Diseño y Programación de Algoritmos para Robot Móviles. Aplicación al robot LEGO-NXT.

PROYECTO FINAL DE CARRERA

REALIZADO POR: ÁNGEL SORIANO VIGUERAS

DIRIGIDO POR: ÁNGEL VALERA FERNÁNDEZ

Índice:

1.- Introducción……7

1.1. Introducción y justificación……7

1.2. Objetivos……7

2.- Desarrollo Teórico……9

2.1. Introducción a la robótica……9

2.2. Tipos de robots……14

2.2.1. Robots Industriales……14

2.2.1.1. Manipuladores……14

2.2.1.2. Robots de repetición o aprendizaje……15

2.2.1.3. Robots con control por computador……16

2.2.2. Robots Inteligentes……17

2.2.3. Micro-robots……18

2.3. Robótica Móvil……19

2.3.1. Clasificación de los robots móviles……19

2.3.1.1. Robots rodantes……19

2.3.1.2. Robots andantes……21

2.3.1.3. Robots reptadores……22

2.3.1.4. Robots nadadores……23

2.3.1.5. Robots voladores……24

2.3.2. Componentes de un robot móvil……25

2.3.2.1. Estructura……25

2.3.2.2. Sensores……26

2.3.2.3. Actuadores……29

2.3.2.4. Sistemas de control……29

2.3.2.5. Alimentación……30

2.3.2.6. Comunicaciones……35
2.4. USB……35
2.5. Comunicaciones inalámbricas……36
2.5.1. Wifi……36
2.5.2. Bluetooth……40
2.5.3. Otras alternativas……41
2.5.3.1. Zigbee……41
2.5.3.2. Infrarrojos……48
2.6. LEGO Nxt……49
2.6.1. Ladrillo NXT……50
2.6.2. Motores……52
2.6.3. Sensores……53
2.6.3.1. Fabricados por LEGO……53
2.6.3.2. Fabricados por terceros……54
2.6.3.3. Personalizados……54
2.6.4. Comunicaciones……54
2.6.5. Software……58
2.6.5.1. LeJOS ……58
2.6.5.2. LEGO NXT-G……58
2.6.5.3. RobotC……59
2.6.5.4. Otras alternativas……60
2.7. Entorno de desarrollo de software……63
2.7.1. Eclipse……63
2.7.2. NetBeans……64
3.- Desarrollo práctico……66
3.1. Preámbulos y configuraciones……66

3.1.1. Montaje del robot NXT……66
3.1.2. Java Developement Kit……66
3.1.3. Instalación del driver USB de Lego……66
3.1.4. Instalación de LejOS NXJ……66
3.1.5. Configuración del entorno de desarrollo (ECLIPSE)……67
3.2. Estudio de comunicaciones mediante Bluetooth……70
3.2.1. Instalación y configuración del dispositivo Bluetooth……70
3.2.2. Configuración de librerías……71
3.2.3. Tiempos de conexión entre dispositivos……71
3.2.4. Estudio de los posibles tipos de datos a transmitir……72
3.3. Logger……72
3.4. Concurrencia en Java. Hilos de ejecución……74
3.5. Simulador de corrección de trayectorias en tiempo real……74
3.6. Estudio del movimiento de los NXT……77
3.6.1. Tratamiento de las funciones de alto nivel……77
3.6.2. Gestión de la trayectoria a bajo nivel……77
3.7. Estrategias del control cinemático……78
3.7.1. Punto descentralizado……78
3.7.2. Persecución pura……80
3.8. The ring: tratamiento de una trayectoria circular, constante, variable en tiempo real y compartida por múltiples robots en un mismo escenario……82
3.8.1. Descripción de la aplicación y características……83
3.8.2. Metodología y desarrollo de la aplicación……83
3.8.3. Sistema de archivos, interfaz y ejecución……86
3.8.4. Problemas encontrados y soluciones……91
3.9. Gestión de la visión……92

3.10. El laberinto: generación de una trayectoria a través del reconocimiento de un escenario mediante una cámara cenital……..94

3.10.1. Descripción……94

3.10.2. Metodología y desarrollo de la aplicación……95

3.10.3. Sistema de archivos, interfaz y ejecución……97

3.10.4. Problemas encontrados y soluciones……..102

4.- Conclusiones y futuros proyectos……107

5.- Bibliografía……109

Anexo 1: Creación de los hilos de ejecución y variables compartidas……111

Anexo2: Establecer una comunicación Bluetooth entre el PC y los robots NXT……..114

Anexo3: Tratamiento de imágenes con OpenCV para Java……117

1.-Introducción

1.1. Introducción y justificación

La búsqueda incesante del hombre por satisfacer su necesidad de comunicación ha sido el impulso que ha logrado la instauración de instrumentos y tecnologías cada día más poderosos y veloces en el proceso comunicativo.

En las últimas décadas la aparición de la comunicación inalámbrica entre dispositivos se ha impuesto con el uso de teléfonos móviles, ordenadores portátiles, PDA's y en general en cualquier dispositivo electrónico. Han aparecido diversos tipos de comunicación como Wifi, Bluetooth o infrarrojos, que aunque hasta el momento no ofrecen prestaciones tan altas como la comunicación por cable, no cabe duda que algún día lo harán.

Al mismo tiempo la tecnología mecánica y electrónica también ha evolucionado y, dejando a un lado los estereotipos de los robots como máquinas que acaban descontrolándose y tomando iniciativa propia, poco a poco se han ido implantando en la sociedad como herramientas que facilitan tareas o resuelven problemas.

Esta tecnología ha llegado incluso al campo de la educación, donde se pueden encontrar juguetes que realmente son auténticos robots en miniatura y que ofrecen tal versatilidad que se hace uso de ellos para llevar acabo investigaciones sobre problemas que podrían abordar más tarde robots más grandes.

La automatización para que estos robots realicen tareas en diferentes entornos con los que tienen que interactuar mediante la integración de visión artificial, reconocimiento de formas, planificación de trayectorias, aún supone un reto en algunos aspectos y ocupa una gran parte de la línea de investigación dentro de la robótica.

Este proyecto engloba el uso de las nuevas tecnologías mecánicas y electrónicas, como los robots de Lego Mindstorms NXT, con el uso de últimas tecnologías en comunicación, como el Bluetooth, y con el estudio de la automatización de seguimiento y generación de trayectorias de vehículos móviles.

1.2. Objetivos

El principal objetivo de este proyecto es tratar de llegar a conocer en profundidad todas las opciones que ofrece el robot Lego Mindstorms NXT con el firmware LejOS y tratar de implementar en base a ese conocimiento diferentes aplicaciones que más tarde se puedan extrapolar a otros robots o escenarios.

Una lista de los objetivos que se han marcado en este proyecto:

- Estudio del firmware LejOS para la plataforma Lego Mindstorm NXT.
- Estudio de la comunicación vía Bluetooth entre robots NXT y el PC.
- Analizar el uso de varios hilos en las comunicaciones.
- Estudio de las estrategias de seguimiento de trayectorias.
- Analizar las posibilidades de captura y reconocimiento de formas en imágenes a través de Java.
- Estudio de algoritmos de generación de trayectorias y generación de splines.
- Desarrollo de aplicaciones que hagan uso de los conocimientos adquiridos

2.- Desarrollo Teórico

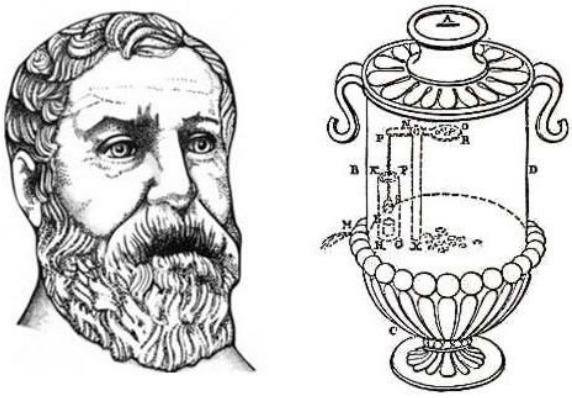
2.1. Introducción a la robótica.

El hombre durante toda su historia ha convivido con el deseo de construir máquinas capaces de realizar tareas que facilitasen su trabajo. Desde hace cientos de años antes de Cristo, ya se intentaban crear dispositivos, denominados artefactos o máquinas, que tuvieran un movimiento sin fin y que no estuvieran controlados ni supervisados por personas.

Los primeros autómatas que aparecen en la historia son ingenios mecánicos, más o menos complicados, que desarrollaban una tarea de forma continua. Sin embargo, las primeras máquinas construidas no tenían una utilidad práctica, sino que su principal objetivo era entretener a sus dueños. Generalmente funcionaban por medio de movimientos ascendentes de aire o agua caliente. El vertido progresivo de un líquido o la caída de un peso provocaba rupturas de equilibrio en diversos recipientes provistos de válvulas; otros mecanismos se basaban en palancas o contrapesos. Mediante sistemas de este tipo se construían pájaros artificiales que podían “cantar" o "volar", o puertas que se abrían solas.

El origen de los autómatas se remonta al Antiguo Egipto, donde las estatuas de algunos de sus dioses despedían fuego por los ojos, poseían brazos mecánicos manejados por los sacerdotes del templo o emitían sonidos cuando los rayos del sol los iluminaba. Estos ingenios pretendían causar temor y respeto entre la gente del pueblo.

Sin embargo, los primeros datos descriptivos acerca de la construcción de un autómata aparecen en el siglo I. El matemático, físico e inventor griego Herón de Alejandría describe múltiples ingenios mecánicos en su libro *Los Autómatas*, por ejemplo aves que vuelan, gorjean y beben.



Herón de Alejandría junto uno de sus inventos, un dispensador de agua sagrada operado por monedas.

Figura 1: Herón junto a uno de sus inventos, un dispensador de agua sagrada operado por monedas:

La Eolípila está considerada como la primera máquina térmica de la historia, pero lamentablemente durante mucho tiempo tan sólo se consideró un juguete sin mayor aplicación. Por otro lado se describen algunos ingenios que sí presentaban una función útil, como en el caso del dispensador de agua por monedas o el de un molino de viento para accionar un órgano (Hydraulis).

Figura 2: La Eolípila

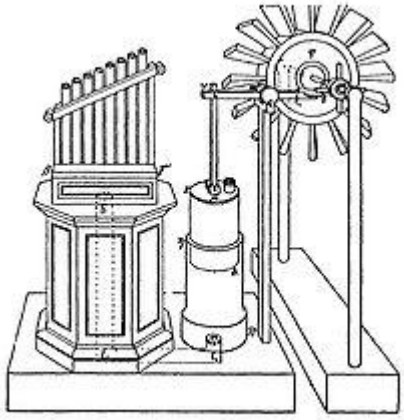


Figura 3: Hydraulis de Ctesibios

Las construcciones de la escuela de Alejandría se extendieron por todo el Imperio Romano y posteriormente por el mundo árabe, siendo estos genuinos aparatos los antecesores de los autómatas actuales.

La cultura árabe heredó y difundió los conocimientos griegos. Al-jazari, uno de los más grandes ingenieros de la historia, fue el creador de muchos inventos de control automático como el cigüeñal o uno de los primeros relojes mecánicos movidos por pesos y agua. Estuvo también muy interesado en la figura del autómata y escribió “El libro del conocimiento de los ingeniosos mecanismos”, obra considerada de las más importantes sobre la historia de la tecnología. Cabe destacar su complejo reloj elefante, animado por seres humanos y animales mecánicos que se movían y marcaban las horas o un autómata con forma humana que servía distintos tipos de bebidas.

Figura4: Reloj elefante, creado por Al-jazari

Del siglo XIII son otros autómatas de los que no han llegado referencias suficientemente bien documentadas, como el *Hombre de Hierro* de Alberto Magno o la *Cabeza Parlante* de Roger Bacon. Otro ejemplo relevante de la época y que aún se conserva en la actualidad es el *Gallo de Estrasburgo*, situado en la catedral de esta misma ciudad, que mueve el pico y las alas al dar las horas.

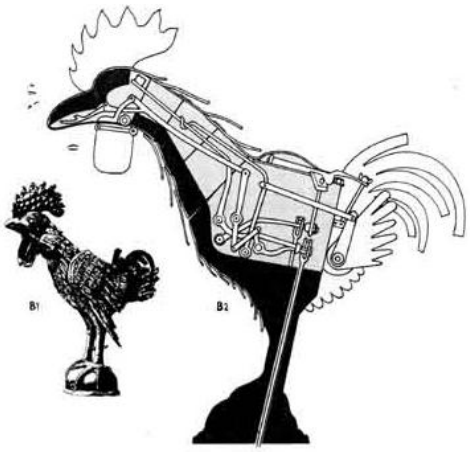


Figura 5: El Gallo de Estrasburgo

Durante los siglos XV y XVI algunas de las figuras más relevantes del Renacimiento también realizaron sus propias aportaciones a la historia de los autómatas. El más famoso quizá sea el *León Mecánico*, desarrollado por Leonardo Da Vinci para el rey Luis XII de Francia, que abría su pecho con la garra y mostraba el escudo de armas del rey. En España, Juanelo Turriano, inventor, arquitecto y relojero real del emperador Carlos V, construyó un autómata de madera con forma de monje conocido como el *Hombre de Palo*, construido con el fin de recolectar limosnas.

En 1738, Jacques de Vaucanson construyó uno de los autómatas más famosos de la historia, el *Pato con aparato digestivo*, considerado su obra maestra. El pato tenía más de 400 partes móviles, y podía batir sus alas, beber agua, digerir grano e incluso defecar. Los alimentos los digería por disolución y eran conducidos por unos tubos hacia el ano, donde había un esfínter que permitía evacuarlos. Vaucanson también construyó otros autómatas, entre los que destaca *El Flautista*, un pastor que tocaba la flauta capaz de tocar hasta 12 melodías diferentes. El ingenio consistía en un complejo mecanismo de aire que causaba el movimiento de todos las diferentes partes del engranaje interno del muñeco. El resultado era una música melodiosa, que tenía el mismo encanto y precisión en las notas que un flautista de carne y hueso.

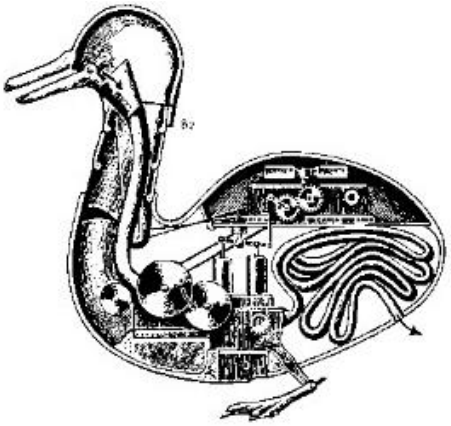


Figura 6: Pato con aparato digestivo

A finales del siglo XVIII y principios del XIX, en plena Revolución Industrial, se desarrollaron diversos ingenios mecánicos utilizados fundamentalmente en la industria textil. Entre ellos se puede citar la hiladora giratoria de Hargreaves (1770), la hiladora mecánica de Crompton (1779), el telar mecánico de Cartwright (1785), el telar de Jacquard (1801), que constituyó con sus tarjetas perforadas los primeros precedentes históricos de las máquinas de control numérico. Más tarde se incorporaron los automatismos en las industrias mineras y metalúrgicas. Pero sin duda, el automatismo que causó mayor impacto por tener un papel fundamental en la Revolución Industrial lo realiza Potter a principios del siglo XVIII, automatizando el funcionamiento de las válvulas en la máquina de vapor atmosférica, creada por el inventor inglés Thomas Newcomen.

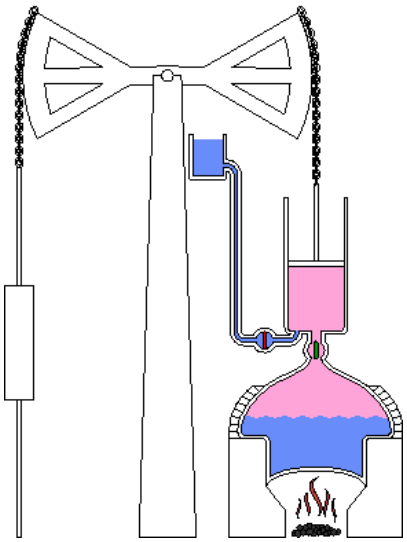


Figura 7: Máquina de vapor atmosférica.

Sin embargo, no fue hasta 1921 cuando se escuchó por primera vez la palabra robot, utilizada por el escritor checo Karel Capek en su obra de teatro *R.U.R (Rossum´s Universal Robots)*. La palabra robot viene del vocablo checo '*Robota*' que significa "trabajo", entendido como servidumbre, trabajo forzado o esclavitud.

Los primeros robots industriales empezaron a producirse a principios de los años 60 y estaban diseñados principalmente para realizar trabajos mecánicos difíciles y peligrosos. Las áreas donde estos robots tuvieron su aplicación fueron trabajos laboriosos y repetitivos, como la carga y descarga de hornos de fundición. En 1961 el inventor norteamericano George Devol patentaba el primer robot programable de la historia, conocido como *Unimate*, estableciendo las bases de la robótica industrial moderna.

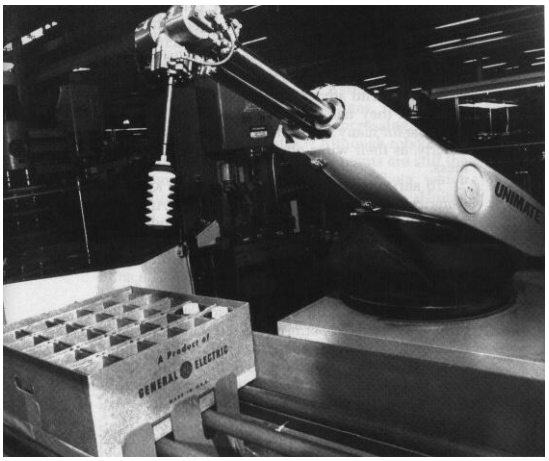


Figura 8: Unimate

Este robot industrial era un manipulador que formaba parte de una célula de trabajo en la empresa de automóviles Ford Motors Company, diseñado para levantar y

apilar grandes piezas de metal caliente, de hasta 225 kg, de una troqueladora de fundición por inyección.

Debido a los continuos avances en la informática y la electrónica, a partir de 1970 fueron desarrollados diversos robots programables, siendo de gran importancia en la industria mecánica, tanto en las líneas de ensamblaje como en aplicaciones como la soldadura o pintura.

En los últimos años, los robots han tomado posición en todas las áreas productivas industriales. La incorporación del robot al proceso productivo ha representado uno de los avances más espectaculares de la edad moderna. En poco más de cuarenta años, se ha pasado de aquellos primeros modelos, rudos y limitados, a sofisticadas máquinas capaces de sustituir al hombre en todo tipo de tareas repetitivas o peligrosas, y además, hacerlo de forma más rápida, precisa y económica que el ser humano. Hoy en día, se calcula que el número de robots industriales instalados en el mundo es de un millón de unidades, unos 20.000 en España, siendo Japón el país más tecnológicamente avanzado, con una media de 322 robots por cada 10.000 trabajadores.

2.2. Tipos de robots

No resulta sencillo hacer una clasificación de tipos de robots, puesto que ningún autor se pone de acuerdo en cuántos y cuáles son los tipos de robots y sus características esenciales.

2.2.1. Robots Industriales

La creciente utilización de robots industriales en el proceso productivo, ha dado lugar al desarrollo de controladores industriales rápidos y potentes, basados en microprocesadores, así como un empleo de servos en bucle cerrado que permiten establecer con exactitud la posición real de los elementos del robot y su desviación o error. Esta evolución ha dado origen a una serie de tipos de robots, que se citan a continuación:

2.2.1.1. Manipuladores

Son sistemas mecánicos multifuncionales, con un sencillo sistema de control, que permite gobernar el movimiento de sus elementos de los siguientes modos:

- **Manual:** Cuando el operario controla directamente la tarea del manipulador.

- **De secuencia fija:** cuando se repite, de forma invariable, el proceso de trabajo preparado previamente.
- **De secuencia variable:** Se pueden alterar algunas características de los ciclos de trabajo

Existen muchas operaciones básicas que pueden ser realizadas de forma óptima mediante manipuladores. Por ello, estos dispositivos son utilizados generalmente cuando las funciones de trabajo son sencillas y repetitivas.

Figura 9: Robot manipulador

2.2.1.2. <u>Robots de repetición o aprendizaje</u>

Son manipuladores que se limitan a repetir una secuencia de movimientos, previamente ejecutada por un operador humano, haciendo uso de un controlador manual o un dispositivo auxiliar. En este tipo de robots, el operario durante la fase de enseñanza se vale de una pistola de programación con diversos pulsadores o teclas, o bien de joysticks, o bien utiliza un maniquí, o desplaza directamente la mano del robot. Actualmente, los robots de aprendizaje son los más conocidos en algunos sectores de la industria, y el tipo de programación que incorporan recibe el nombre de "*gestual*".

Figura 10: Robot Pingüino de aprendizaje

2.2.1.3. Robots con control por computador

Son manipuladores o sistemas mecánicos multifuncionales, controlados por un computador, que habitualmente suele ser un microordenador. El control por computador dispone de un lenguaje específico de programación, compuesto por varias instrucciones adaptadas al hardware del robot, con las que se puede diseñar un programa de aplicación utilizando solo el ordenador. A esta programación se le denomina "*textual*" y se crea sin la intervención del manipulador.

Las grandes ventajas que ofrece este tipo de robots, hacen que se vayan imponiendo en el mercado rápidamente, lo que exige la preparación urgente de personal cualificado, capaz de desarrollar programas de control que permitan el manejo del robot.

Figura 11: Robot FANUC

2.2.2. Robots Inteligentes

Son similares a los del grupo anterior, pero tienen la capacidad de poder relacionarse con el mundo que les rodea a través de sensores y de tomar decisiones en función de la información obtenida en tiempo real. De momento, son muy poco conocidos en el mercado y se encuentran en fase experimental, donde grupos de investigadores se esfuerzan por hacerlos más efectivos, al mismo tiempo que más económicamente asequibles. El reconocimiento de imágenes y algunas técnicas de inteligencia artificial son los campos que más se están estudiando para su posible aplicación en estos robots.

Figura 12: Robot ASIMO de Honda

2.2.3. Micro-robots

Con fines educacionales, de entretenimiento o investigación, existen numerosos robots de formación o micro-robots a un precio muy asequible, cuya estructura y funcionamiento son similares a los de aplicación industrial.

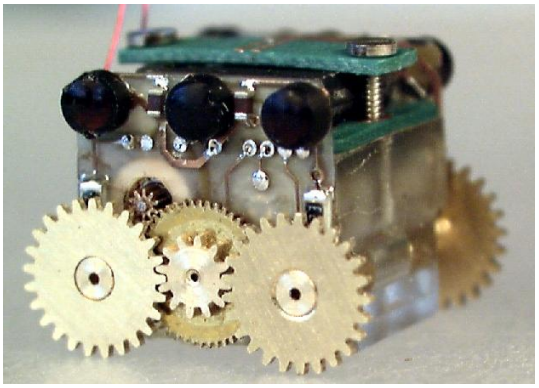


Figura 13: Robot Smoovy

Estos robots que un día se hicieron un hueco en universidades y centros de investigación, puesto que eran una forma económica de experimentar con múltiples tareas robóticas, hoy en día se pueden encontrar en centros docentes de todo tipo, incluidas escuelas de primaria e institutos. El personal de dichos centros ha apostado por este tipo de robots para estimular el interés de sus alumnos por la ciencia y la tecnología y los resultados son como se ha podido observar altamente satisfactorios.

Figura 14: Robot LEGO

2.3. Robótica Móvil

En el apartado anterior se ha realizado un desglose de los diferentes tipos de robots existentes atendiendo a su aplicación, pero más allá de este aspecto práctico hay otro hecho característico de los robots modernos que les confiere un mayor grado de libertad y utilidad. Esta característica es el movimiento en el espacio físico, es decir, la posibilidad de desplazarse por el entorno para observarlo e interactuar con él, y de esta forma emular con mayor fidelidad las funciones y capacidades de los seres vivos.

2.3.1. Clasificación de los robots móviles

De la misma forma que se ha descrito en el apartado anterior en base a diferentes criterios se puede establecer una taxonomía dentro del colectivo de los robots móviles. Si por ejemplo se atiende a sus características estructurales y funcionales se puede establecer la siguiente clasificación:

2.3.1.1. Robots rodantes

Son aquellos que, como su nombre indica, se desplazan haciendo uso de ruedas, generalmente montadas por pares en una configuración 2+2 como las de un vehículo por mera simplicidad. Habitualmente solo dos de sus ruedas presentan tracción y otras dos dirección, de forma que sea posible maniobrar el robot con un solo servomotor.

Figura 25: Robot rodante dotado de 4 ruedas en configuración 2+2

También es frecuente encontrar distribuciones de ruedas montadas en modo triciclo, donde una rueda sirve para la dirección y las otras dos aportan la tracción. Otra opción es que la tercera rueda simplemente sea una rueda 'loca' y las otras dos aporten tanto la tracción como la dirección, mediante el método de las orugas tratado más adelante.

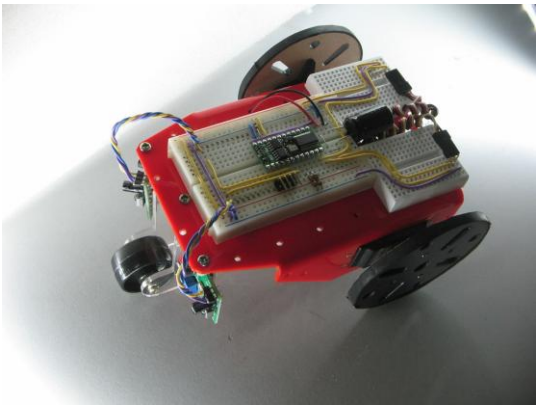


Figura 36: Robot rodante con las ruedas en configuración triciclo

Existen algunos casos especiales en los que se usan otras configuraciones que dotan al robot de mejor adaptación a terrenos difíciles. En estos casos los algoritmos de control de movimiento adquieren una mayor complejidad, proporcional al número de elementos direccionables de forma independiente.

Figura 47: Robot rodante de 6 ruedas con dirección independiente frontal y trasera

Por último cabría considerar a los robots con orugas como un tipo de robot rodante en el que se substituyen las ruedas por un mecanismo de oruga para la tracción. La dirección se consigue parando una de las orugas o haciéndolas girar en sentido contrario.

Figura 58: Robot rodante dotado de orugas

2.3.1.2. Robots andantes

Respecto a los robots construidos a imagen y semejanza humana, con dos piernas, las técnicas de control necesarias son varias, pero todas ellas hacen uso de compLeJOS algoritmos para poder mantener el equilibrio y caminar correctamente. Todos ellos son capaces de caminar bien sobre suelos planos y subir escaleras en algunos casos, pero no están preparados para caminar en suelos irregulares.

Algunos incluso pueden realizar tareas como bailar, luchar o practicar deportes, pero esto requiere una programación sumamente compleja que no siempre está a la altura del hardware del robot y de su capacidad de procesamiento.

Figura 19: Robot humanoide Robonova.

2.3.1.3. Robots reptadores

Una clase curiosa de robots, creados basándose en animales como las serpientes, su forma de desplazarse es también una imitación de la usada por estos animales. Están formados por un número elevado de secciones que pueden cambiar de tamaño o posición de forma independiente de las demás pero coordinada, de forma que en conjunto provoquen el desplazamiento del robot.

Figura 206: Robot reptador

2.3.1.4. Robots nadadores

Estos robots son capaces de desenvolverse en el medio acuático, generalmente enfocados a tareas de exploración submarina en zonas donde no es posible llegar pos ser de difícil acceso o estar a profundidades que el cuerpo humano no tolera.

Figura 21: Robot pez

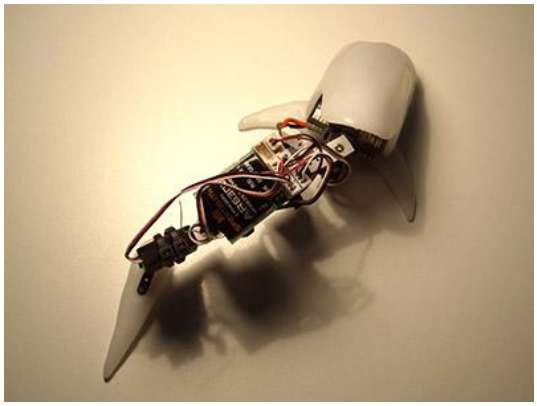


Figura 22: Robot ballena

Aparte de lo puramente anecdótico, se ha demostrado que la estructura corporal de los peces así como el movimiento que realizan durante su desplazamiento en el agua, es uno de los métodos más óptimos de movimiento submarino dado que aprovecha la energía de forma muy eficiente y permite mayor control en la navegación, produciendo mucho menos ruido y turbulencias.

Es por todo esto que se está tendiendo a estudiar y emular en lo posible el comportamiento de estos animales a la hora de crear nuevos robots subacuáticos.

Figura 7: Atún robótico

2.3.1.5. Robots voladores

Conquistados los dominios del mar y la tierra solo queda una meta por alcanzar en el mundo de la robótica, ser capaces de poner robots en el cielo. Para ello y por el momento existen dos aproximaciones, en función de su principio de vuelo y estructura:

- **Helicópteros:** habitualmente helicópteros RC convencionales a los que se les añade la electrónica necesaria para tener visión artificial y capacidad de toma de decisiones autónoma.

Figura 8: Helicóptero robótico

- **Drones o aviones no tripulados:** actualmente en uso por el ejército de EEUU para tareas de logística en operaciones militares, apoyo cartográfico así como tareas de espionaje. Aunque no es habitual pueden estar dotados tanto de contramedidas para repeler agresiones como de armamento para realizar ataques.

Figura 9: Drone (robot volador militar no tripulado)

2.3.2. Componentes de un robot móvil

Vistos los principales tipos de robots móviles que se construyen en la actualidad, a continuación se detallan las partes constituyentes de los robots, tanto estructurales, como mecánicas y electrónicas.

2.3.2.1. Estructura

La estructura es el esqueleto, el soporte fundamental que constituye tanto la forma como la funcionalidad del robot. Sirve de sujeción para toda la electrónica, sensores, actuadores y también es parte del aparato motriz como prolongación de los actuadores. Está formada generalmente por una mezcla de partes rígidas y flexibles, fijas y móviles y entre sus materiales destacan los plásticos, metales y aleaciones resistentes a la par que ligeras como la fibra de carbono o derivados del aluminio.

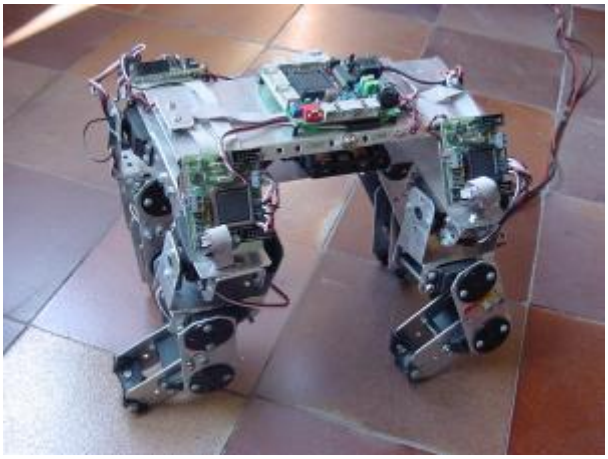


Figura 10: Estructura básica fabricada en aluminio para un robot cuadrúpedo.

Determina el medio en el que se va a poder desenvolver el robot, así como las actividades que será capaz de realizar. También protege partes sensibles de la electrónica de golpes, polvo, agua y otros agentes externos. Como ya se ha comentado, debe ser un compromiso entre resistencia y ligereza, adaptándose de la mejor manera posible al tipo de tareas para las que se diseña el robot que va a poseer dicha estructura.

2.3.2.2. Sensores

Estos elementos son los encargados de adquirir información del entorno y transmitirla a la unidad de control del robot. Una vez esta es analizada, el robot realiza la acción correspondiente a través de sus actuadores.

Los sensores constituyen el sistema de percepción del robot, es decir, facilitan la información del mundo real para que el robot la interprete. Los tipos de sensores más utilizados son los siguientes:

- **Sensor de proximidad:** Detecta la presencia de un objeto ya sea por rayos infrarrojos, por sonar, magnéticamente o de otro modo.

Figura 115: Sensores de proximidad

• **Sensor de Temperatura:** Capta la temperatura del ambiente, de un objeto o de un punto determinado.

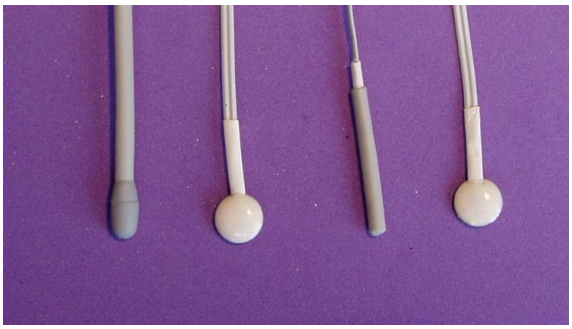


Figura 26: Sensores de temperatura

• **Sensor magnéticos:** Captan variaciones producidas en campos magnéticos externos. Se utilizan a modo de brújulas para orientación geográfica de los robots.

Figura 27: Sensores magnéticos

• **Sensores táctiles, piel robótica:** Sirven para detectar la forma y el tamaño de los objetos que el robot manipula. La piel robótica se trata de un conjunto de sensores de presión montados sobre una superficie flexible.

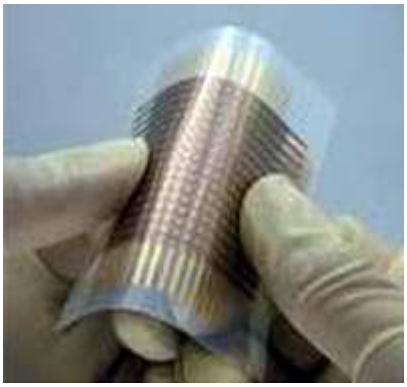


Figura 28: Sensor táctil flexible

• **Sensores de iluminación:** Capta la intensidad luminosa, el color de los objetos, etc. Es muy útil para la identificación de objetos. Es parte de la visión artificial y en numerosas ocasiones son cámaras.

Figura 29: Sensores de luz

• **Sensores de velocidad, de vibración (Acelerómetro) y de inclinación:** Se emplean para determinar la velocidad de actuación de las distintas partes móviles del propio robot o cuando se produce una vibración. También se detecta la inclinación a la que se encuentra el robot o una parte de él.

Figura 12: Sensores de velocidad de reluctacia variable

• **Sensores de fuerza:** Permiten controlar la presión que ejerce la mano del robot al coger un objeto.

Figura 31: Sensor de presión

• **Sensores de sonido:** Micrófonos que permiten captar sonidos del entorno.

Figura 13: Micrófonos

- **Micro interruptores:** Muy utilizados para detectar finales de carrera.

Figura 33: Diferentes tipos de micro interruptores

2.3.2.3. Actuadores

Los actuadores son los sistemas de accionamiento que permiten el movimiento de las articulaciones del robot. Se clasifican en tres grupos, dependiendo del tipo de energía que utilicen:

- **Hidráulicos:** se utilizan para manejar cargas pesadas a una gran velocidad. Sus movimientos pueden ser suaves y rápidos.

- **Neumáticos:** son rápidos en sus respuestas, pero no soportan cargas tan pesadas como los hidráulicos.

- **Eléctricos**: son los más comunes en los robots móviles. Un ejemplo son los motores eléctricos, que permiten conseguir velocidades y precisión necesarias.

Ejemplos de actuadores son motores, relés y contadores, electro válvulas, pinzas, etc.

2.3.2.4. Sistemas de control

El control de un robot puede realizarse de muchas maneras, pero generalmente se realiza por medio de un ordenador industrial altamente potente, también conocido como unidad de control o controlador. El controlador se encarga de almacenar y procesar la información de los diferentes componentes del robot industrial.

La definición de un sistema de control es la combinación de componentes que actúan juntos para realizar el control de un proceso. Este control se puede hacer de forma continua, es decir en todo momento o de forma discreta, es decir cada cierto tiempo. Si el sistema es continuo, el control se realiza con elementos continuos. En

cambio, cuando el sistema es discreto el control se realiza con elementos digitales, como el ordenador, por lo que hay que digitalizar los valores antes de su procesamiento y volver a convertirlos tras el procesamiento.

Existen dos tipos de sistemas, sistemas en lazo abierto y sistemas en lazo cerrado.

- **Sistemas en bucle abierto**: son aquellos en los que la salida no tiene influencia sobre la señal de entrada.

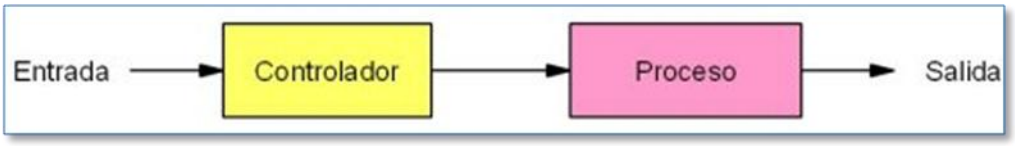


Figura 34: Esquema de un controlador en bucle abierto

- **Sistemas en bucle cerrado:** son aquellos en los que la salida influye sobre la señal de entrada.

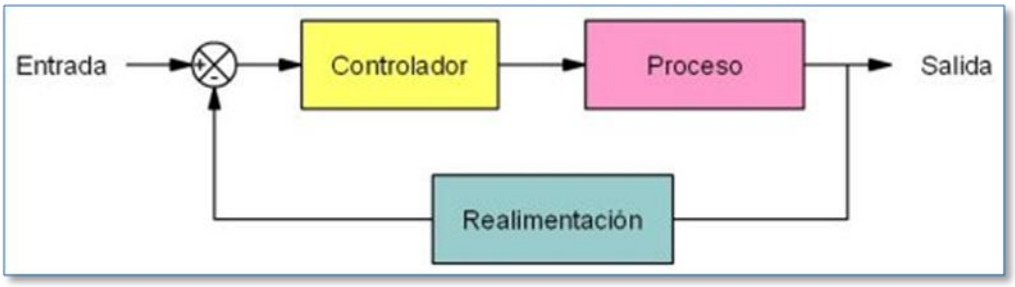


Figura 35: Esquema de un controlador en bucle cerrado

- **Sistemas discretos:** son aquellos que realizan el control cada cierto tiempo. En la actualidad se utilizan sistemas digitales para el control, siendo el ordenador el más utilizado, por su fácil programación y versatilidad.

Generalmente, el control en los robots se realiza mediante sistemas discretos en lazo cerrado, realizados por computador. El ordenador procesa la información captada por los sensores y activa los actuadores en intervalos lo más cortos posibles, del orden de milisegundos.

2.3.2.5. Alimentación

Parte fundamental para garantizar la autonomía del robot, dado que al ser móvil generalmente no va a poder tener energía externa, sólo contará con las reservas internas que pueda transportar, salvo en el caso de que se usen placas solares. Son muchas y muy diversas las formas en que el robot puede almacenar y transportar energía, por lo que veremos las principales:

- **Baterías:** sin duda la forma más comúnmente utilizada como fuente de alimentación en todo tipo de robots. Son baratas, fiables y se pueden encontrar en cientos de formatos, voltajes y dimensiones.

Figura 14: Diferentes tipos de baterías de uso doméstico

Desde la pila de botón más pequeña y ligera para alimentar un micro robot podemos ir a pesadas pero potentes baterías de plomo usadas en vehículos pesados que necesitan un aporte importante de energía.

Otro aspecto importante a tener en cuenta es el formato y el conexionado de las pilas. Habitualmente encontramos las baterías recargables tanto sueltas como en packs preparados para soldar o para conectar directamente a un PCB. La elección de un formato u otro es más relevante de lo que parece.

Por ejemplo, un robot móvil todo terreno que lleve las baterías de forma individual en un porta-pilas no sería nada extraño que con el movimiento alguna pila dejase de hacer contacto o incluso se saliera de su sitio. Y en el otro extremo, si las pilas estuvieran soldadas habría serios problemas para reemplazarlas con facilidad, por no hablar de la dependencia a la hora de cargarlas que deberá ser realizado exclusivamente en el robot, teniendo que añadir los mecanismos necesarios para ello.

Parece por tanto que lo más adecuado es usar pilas en packs con conectores, que son las que mayor flexibilidad aportan.

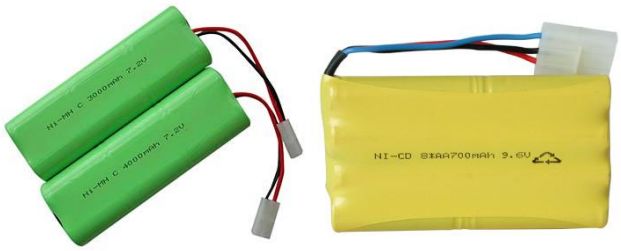


Figura 157: Diferentes tipos de baterías

Si el presupuesto no es un problema se pueden usar tecnologías más avanzadas como las baterías de ion de litio que últimamente están sufriendo una gran expansión gracias a tecnologías móviles de otra índoles como ordenadores portátiles y teléfonos móviles.

Figura 3816: Amplio surtido de baterías de Ion de Litio

La diferencia de rendimiento de unos tipos de pilas respecto a otros es notable, así como su tamaño y precio, por lo tanto es importante tener claros los requerimientos del robot respecto a estos aspectos antes de decidir qué tipo de batería se va a utilizar para alimentarlo.

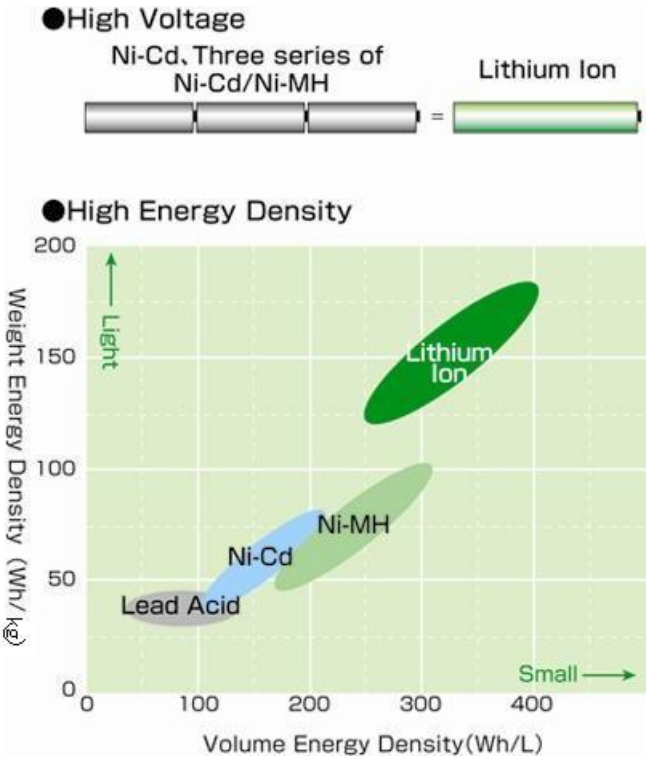


Figura 3917: Gráfico comparativo del rendimiento de diferentes tipos de baterías

• **Bombonas de aire:** usadas tanto para el movimiento de actuadores neumáticos como para la propulsión de motores de aire. Este tipo de motores tiene una larga historia, pese a que su difusión es reducida. Su principio es equivalente al de los motores de explosión, con la salvedad que en lugar de producir la combustión de gasolina y oxígeno para generar los gases que impulsan los pistones, se usa directamente aire almacenado en un depósito a alta presión.

Figura 4018: Diferentes tipos de bombonas de aire comprimido

• **Baterías inerciales:** este curioso mecanismo tampoco tiene una difusión amplia en el terreno de los robots móviles, pero si tiene aplicación e entornos donde la alimentación es crítica, como centros de proceso de datos. Fue probada en vehículos durante sus albores, concretamente en autobuses urbanos de Zúrich durante los años 50 y algunas locomotoras de tren durante los 70, pero no se ha extendido su uso de forma masiva.

El principio de funcionamiento es sencillo: se acelera un rotor (habitualmente mediante electricidad) confiriéndole así una elevada energía rotacional, típicamente de varias decenas de miles de rpm. Posteriormente se decelera poco a poco, recuperando así la energía eléctrica de nuevo.

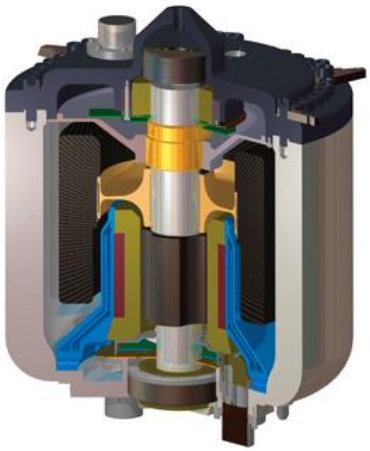


Figura 4119: Sección de una batería inercial

La clave del aprovechamiento energético, que oscila entre el 90% y el 98%, consiste en reducir al máximo el rozamiento. Esto se consigue mediante usando un rotor de carbono (que presenta menos resistencia que el acero) estabilizado magnéticamente en un entorno controlado, generalmente un contenedor sellado tras realizarle el vacío.

- **Células de hidrogeno:** Otra tecnología emergente que poco a poco está demostrando su validez son las baterías formadas a base de células que usan el hidrógeno como combustible para producir electricidad.

Su funcionamiento se basa en una reacción química al igual que en las baterías tradicionales pero a diferencia de ellas, aquí no se producen cambios de estado en los electrodos o en los reactantes. Simplemente se consumen los reactantes y la reacción se produce prácticamente de forma indefinida mientras haya suficiente combustible para sustentar el proceso.

Figura 202: Célula de hidrógeno

Las principales ventajas son su elevado rendimiento así como su larga vida, que va asociada a la práctica ausencia de mantenimiento que necesitan. En contraposición el combustible a día de hoy todavía es relativamente caro y difícil de conseguir aunque esta situación mejorará cuando haya una mayor demanda de estas células.

- **Biomasa:** Una fuente de energía muy interesante y que aun se encuentra por explotar. Pequeñas cantidades de basura orgánica permitirían alimentar durante horas un dispositivo dotado de esta tecnología. A parte del ahorro económico que supone, tiene especial interés en zonas sin acceso directo a fuentes de energía o en momentos de conflicto militar donde frecuentemente escasean los suministros energéticos.

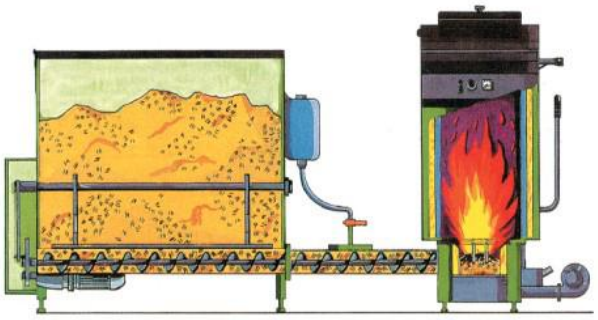


Figura 43: Caldera de biomasa.

- **Células nucleares:** Se basan en las emisiones de un isótopo radioactivo para obtener energía. Hay diversas técnicas para realizar la conversión de las radiaciones en energía eléctrica aprovechable, pero el resultado es el mismo. Poseen una larga vida útil pero su peligrosidad desaconseja su uso.

- **Motor de combustión:** El más conocido entre los sistemas de propulsión. Se usan motores muy similares a los de los vehículos, sobre todo los que llevan las motocicletas. Como combustible una mezcla de aceite y gasolina u otros elementos volátiles como el etanol. Se pueden utilizar tanto para obtener energía mecánica como energía eléctrica si añadimos un generador.

2.3.2.6. Comunicaciones

Dado que este es uno de los aspectos sobre los que versan las experiencias realizadas en este proyecto, se le dedicara el apartado siguiente a este tema, de forma que se traten en profundidad las características más relevantes:

2.4. USB

El Universal Serial Bus (bus universal en serie) o Conductor Universal en Serie (CUS), abreviado comúnmente USB, es un puerto de comunicación de periféricos a un computador. Fue creado en 1996 con la finalidad de eliminar la necesidad de adquirir tarjetas separadas para poner en los puertos de bus ISA o PCI y mejorar las capacidades plug-and-play permitiendo a esos dispositivos ser conectados o desconectados sin necesidad de reiniciar el sistema. Sin embargo, en aplicaciones donde se necesita ancho de banda para grandes transferencias de datos los buses PCI o PCIe salen ganando e igual sucede si la aplicación requiere robustez industrial.

Se pueden clasificar en cuatro tipos según su velocidad de transferencia de datos:

- Baja velocidad (1.0): Tasa de transferencia de hasta 1,5 Mbps (192KB/s). Utilizado en su mayor parte por dispositivos de interfaz humana como los teclados, ratones e incluso artículos del hogar.

- Velocidad completa (1.1): Tasa de transferencia de hasta 12Mbps (1,5MB/s), según el estándar, pero se dice en fuentes independientes que habría que realizar nuevamente las mediciones.

- Alta velocidad (2.0): Tasa de transferencia de hasta 480 Mbps (60MB/s) pero por lo general de hasta 125Mbps (16MB/s). Está presente casi en el 99% de los ordenadores actuales. El cable USB 2.0 dispone de cuatro líneas, una para datos, una para corriente y otra de toma de tierra.

- Super alta velocidad (3.0): Actualmente se encuentra en fase experimental y tiene una tasa de transferencia de hasta 4.8 Gbps (600MB/s). Esta especificación será diez veces más veloz que la anterior 2.0 y prevé su lanzamiento a corto plazo. Se han incluido cinco conectores extra, desechando el conector de fibra óptica propuesto inicialmente y será compatible con los estándares anteriores.

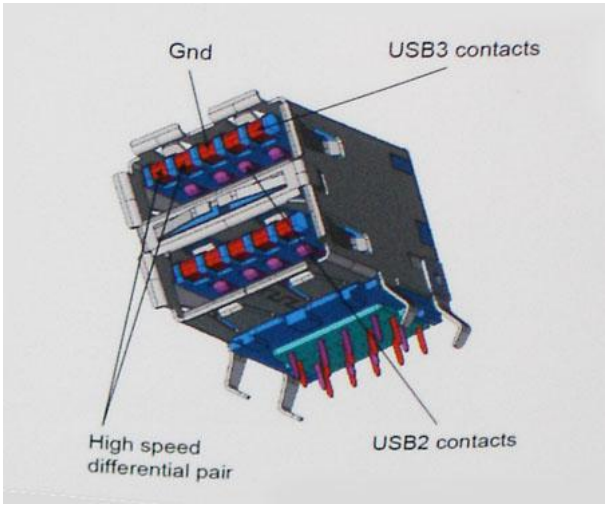


Figura 44: Prototipo de USB 3.0

2.5. Comunicaciones inalámbricas

2.5.1. Wifi

WLAN (Wireless Local Area Network, en inglés) es un sistema de comunicación de datos inalámbrico flexible, muy utilizado como alternativa a las redes LAN cableadas o como extensión de éstas. Utiliza tecnología de radiofrecuencia que permite mayor movilidad a los usuarios al minimizar las conexiones cableadas. Las WLAN van adquiriendo importancia en muchos campos, como almacenes o para manufactura, en los que se transmite la información en tiempo real a una terminal central. También son

muy populares en los hogares para compartir el acceso a Internet entre varias computadoras.

Características

- Movilidad: permite transmitir información en tiempo real en cualquier lugar de la organización o empresa a cualquier usuario. Esto supone mayor productividad y posibilidades de servicio.
- Facilidad de instalación: al no usar cables, se evitan obras para tirar cable por muros y techos, mejorando así el aspecto y la habitabilidad de los locales, y reduciendo el tiempo de instalación. También permite el acceso instantáneo a usuarios temporales de la red.
- Flexibilidad: puede llegar donde el cable no puede, superando mayor número de obstáculos, llegando a atravesar paredes. Así, es útil en zonas donde el cableado no es posible o es muy costoso: parques naturales, reservas o zonas escarpadas.

Inicios

Los pioneros en el uso de redes inalámbricas han sido los radioaficionados mediante sus emisoras, que ofrecen una velocidad de 9600 bps. Pero si hablamos propiamente de redes inalámbricas debemos remontarnos al año 1997, en el que el organismo regulador IEEE (Institute of Electronics and Electrical Engineer) publicó el estándar 802.11 (802 hace referencia al grupo de documentos que describen las características de las LAN) dedicado a redes LAN inalámbricas.

Dentro de este mismo campo y anteriormente, en el año 1995, tenemos la aparición de Bluetooth, una tecnología de Ericsson con el objetivo de conectar mediante ondas de radio los teléfonos móviles con diversos accesorios. Al poco tiempo se generó un grupo de estudio formado por fabricantes que estaban interesados en esta tecnología para aplicarla a otros dispositivos, como PDAs, terminales móviles o incluso electrodomésticos.

Pero el verdadero desarrollo de este tipo de redes surgió a partir de que la FCC, el organismo americano encargado de regular las emisiones radioeléctricas, aprobó el uso civil de la tecnología de transmisiones de espectro disperso (SS o spread spectrum, en inglés), pese a que en un principio lo prohibió por el uso ampliado del espectro. Dicha tecnología ya se usaba en ámbitos militares desde la Segunda Guerra Mundial debido a sus extraordinarias características en cuanto a la dificultad de su detección y su tolerancia a interferencias externas.

A pesar, de que como hemos visto, esta tecnología ya tiene una antigüedad de más de diez años, no ha sido hasta ahora cuando este tipo de redes se ha desarrollado eficazmente debido a la disminución de precios de los dispositivos que la integran. En

la actualidad cada vez más se encuentran equipos que pueden competir en precios con los modelos para redes cableadas.

Cómo trabajan las redes WLAN

Se utilizan ondas de radio para llevar la información de un punto a otro sin necesidad de un medio físico guiado. Al hablar de ondas de radio nos referimos normalmente a portadoras de radio, sobre las que va la información, ya que realizan la función de llevar la energía a un receptor remoto. Los datos a transmitir se superponen a la portadora de radio y de este modo pueden ser extraídos exactamente en el receptor final.

A este proceso se le llama modulación de la portadora por la información que está siendo transmitida. Si las ondas son transmitidas a distintas frecuencias de radio, varias portadoras pueden existir en igual tiempo y espacio sin interferir entre ellas. Para extraer los datos el receptor se sitúa en una determinada frecuencia, frecuencia portadora, ignorando el resto. En una configuración típica de LAN sin cable los puntos de acceso (transceiver) conectan la red cableada de un lugar fijo mediante cableado normalizado. El punto de acceso recibe la información, la almacena y la transmite entre la WLAN y la LAN cableada. Un único punto de acceso puede soportar un pequeño grupo de usuarios y puede funcionar en un rango de al menos treinta metros y hasta varios cientos. El punto de acceso (o la antena conectada al punto de acceso) es normalmente colocado en alto pero podría colocarse en cualquier lugar en que se obtenga la cobertura de radio deseada. El usuario final accede a la red WLAN a través de adaptadores. Estos proporcionan una interfaz entre el sistema de operación de red del cliente y las ondas, mediante una antena.

La naturaleza de la conexión sin cable es transparente a la capa del cliente.

Configuraciones de red para radiofrecuencia

Pueden ser de muy diversos tipos y tan simples o complejas como sea necesario. La más básica se da entre dos ordenadores equipados con tarjetas adaptadoras para WLAN, de modo que pueden poner en funcionamiento una red independiente siempre que estén dentro del área que cubre cada uno. Esto es llamado red de igual a igual (peer to peer). Cada cliente tendría únicamente acceso a los recursos del otro cliente pero no a un servidor central. Este tipo de redes no requiere administración o preconfiguración.

Instalando un Punto de Acceso se puede doblar la distancia a la cual los dispositivos pueden comunicarse, ya que estos actúan como repetidores. Desde que el punto de acceso se conecta a la red cableada cualquier cliente tiene acceso a los recursos del servidor y además gestionan el tráfico de la red entre los terminales más próximos.

Cada punto de acceso puede servir a varias máquinas, según el tipo y el número de transmisiones que tienen lugar. Existen muchas aplicaciones en el mundo real con un rango de 15 a 50 dispositivos cliente con un solo punto de acceso.

Los puntos de acceso tienen un alcance finito, del orden de 150 m en lugares u zonas abiertas. En zonas grandes como por ejemplo un campus universitario o un edificio es probablemente necesario más de un punto de acceso. La meta es cubrir el área con células que solapen sus áreas de modo que los clientes puedan moverse sin cortes entre un grupo de puntos de acceso.

Esto es llamado roaming, mediante el cual el diseñador de la red puede elegir usar un Punto de Extensión (EPs) para aumentar el número de puntos de acceso a la red, de modo que funcionan como tales pero no están enganchados a la red cableada como los puntos de acceso.

Los puntos de extensión funcionan como su nombre indica: extienden el alcance de la red retransmitiendo las señales de un cliente a un punto de acceso o a otro punto de extensión. Los puntos de extensión pueden encadenarse para pasar mensajes entre un punto de acceso y clientes lejanos de modo que se construye un puente entre ambos.

Uno de los últimos componentes a considerar en el equipo de una WLAN es la antena direccional. Por ejemplo: si se quiere una LAN sin cable a otro edificio a 1 km de distancia. Una solución puede ser instalar una antena en cada edificio con línea de visión directa. La antena del primer edificio está conectada a la red cableada mediante un punto de acceso. Igualmente en el segundo edificio se conecta un punto de acceso, lo cual permite una conexión sin cable en esta aplicación.

Asignación de Canales

Los estándares 802.11b y 802.11g utilizan la banda de 2.4 – 2.5 Ghz. En esta banda, se definieron 11 canales utilizables por equipos WIFI, los que pueden configurarse de acuerdo a necesidades particulares. Sin embargo, los 11 canales no son completamente independientes (canales contiguos se superponen y se producen interferencias) y en la práctica sólo se pueden utilizar 3 canales en forma simultánea (1, 6 y 11). Esto es correcto para USA y muchos países de América Latina, pues en Europa, el ETSI ha definido 13 canales. En este caso, por ejemplo en España, se pueden utilizar 4 canales no-adyacentes (1, 5, 9 y 13). Esta asignación de canales usualmente se hace sólo en el Access Point, pues los “clientes” automáticamente detectan el canal, salvo en los casos en que se forma una red “Ad-Hoc” o punto a punto cuando no existe Access Point.

Seguridad

Uno de los problemas de este tipo de redes es precisamente la seguridad ya que cualquier persona con una terminal inalámbrica podría comunicarse con un punto de acceso privado si no se disponen de las medidas de seguridad adecuadas. Dichas medidas van encaminadas en dos sentidos: por una parte está el cifrado de los datos que se transmiten y en otro plano, pero igualmente importante, se considera la autenticación entre los diversos usuarios de la red. En el caso del cifrado se están realizando diversas investigaciones ya que los sistemas considerados inicialmente se han conseguido descifrar. Para la autenticación se ha tomado como base el protocolo de verificación EAP (Extesible Authentication Protocol), que es bastante flexible y permite el uso de diferentes algoritmos.

2.5.2. Bluetooth

Bluetooth proviene de la palabra escandinava "*Blåtand"* que significa "hombre de tez oscura" pero en los tiempos que corren el significado original se ha perdido y ahora se asocia a las comunicaciones inalámbricas, un estándar global que posibilita la transmisión de voz, imágenes y en general datos entre diferentes dispositivos en un radio de corto alcance y lo que le hace más atractivo, muy bajo coste.

Los principales objetivos que este estándar quiere lograr son:

- Facilitar las comunicaciones entre equipos.
- Eliminar cables y conectores entre aquéllos.
- Facilitar el intercambio de datos entre los equipos.

Bluetooth funciona bajo radio frecuencias pudiendo atravesar diferentes obstáculos para llegar a los dispositivos que tenga a su alcance.

Opera bajo la franja de frecuencias 2.4 – 2.48 GHz o como también es conocida como "Banda ISM" que significa "Industrial, Scientific and Medical" que es una banda libre para usada para investigar por los tres organismos anteriores. Pero esto tiene sus consecuencias, ya que al ser libre puede ser utilizada por cualquiera y para ello, para evitar las múltiples interferencias que se pudieran introducir (microondas, WLANs, mandos, etc.) Bluetooth utiliza una técnica denominada salto de frecuencias.

El funcionamiento es ir cambiando de frecuencia y mantenerse en cada una un "slot" de tiempo para después volver a saltar a otra diferente. Es conocido que entre salto y salto el tiempo que transcurre es muy pequeño, concretamente unos 625 microsegundos con lo que al cabo de un segundo se puede haber cambiado 1600 veces de frecuencia.

Cuando coinciden más de un dispositivo bluetooth en un mismo canal de transmisión se forma lo que se llaman "piconets" que son redes donde hay un maestro

que es el que gestiona la comunicación de la red y establece su reloj y unos esclavos que escuchan al maestro y sincronizan su reloj con el del maestro.

Dicho alcance puede variar según la potencia a la que se transmite (a mayor potencia mayor consumo y menor autonomía para el dispositivo) y el número de repetidores que haya por el medio. Así el alcance puede estar entre los 10 y 100 metros de distancia (los repetidores provocan la introducción de distorsión que puede perjudicar los datos transmitidos).

Bluetooth puede conectar muchos tipos de aparatos sin necesidad de un solo cable, aportando una mayor libertad de movimiento. Por esta razón ya se ha convertido en una norma común mundial para la conexión inalámbrica. En el futuro, es probable que sea una norma utilizada en millones de teléfonos móviles, PC, ordenadores portátiles y varios tipos de aparatos electrónicos, como por ejemplo:

- Domótica (activación de alarmas, subida de persianas, etc.).
- Sector automovilístico (comunicación con otros vehículos).
- Medios de pago.

En una comunicación Bluetooth se pueden alcanzar tasas de transmisión de datos de 720 kbps (1 Mbps de capacidad bruta) con un rango óptimo de 10 metros (como se ha comentado antes 100 metros con repetidores).

2.5.3. Otras alternativas

2.5.3.1. Zigbee

ZigBee es el nombre de la especificación de un conjunto de protocolos de alto nivel de comunicación inalámbrica para su utilización con radios digitales de bajo consumo, basada en el estándar IEEE 802.15.4 de redes inalámbricas de área personal (wireless personal area network, WPAN). Su objetivo son las aplicaciones que requieren comunicaciones seguras con baja tasa de envío de datos y maximización de la vida útil de sus baterías.

En principio, el ámbito donde se prevé que esta tecnología cobre más fuerza es en domótica, como puede verse en los documentos de la ZigBee Alliance, en las referencias bibliográficas que se dan más abajo es el documento «ZigBee y Domótica». La razón de ello son diversas características que lo diferencian de otras tecnologías:

- Su bajo consumo
- Su topología de red en malla
- Su fácil integración (se pueden fabricar nodos con muy poca electrónica).

Visión general

La relación entre IEEE 802.15.4-2003 y ZigBee es parecida a la existente entre IEEE 802.11 y Wi-Fi Alliance. La especificación 1.0 de ZigBee se aprobó el 14 de diciembre de 2004 y está disponible a miembros del grupo de desarrollo (ZigBee Alliance). Un primer nivel de suscripción, denominado adopter, permite la creación de productos para su comercialización adoptando la especificación por 3500 dólares anuales. Esta especificación está disponible al público para fines no comerciales en la petición de descarga. La revisión actual de 2006 se aprobó en diciembre de dicho año.

ZigBee utiliza la banda ISM para usos industriales, científicos y médicos; en concreto, 868 MHz en Europa, 915 en Estados Unidos y 2,4 GHz en todo el mundo. Sin embargo, a la hora de diseñar dispositivos, las empresas optarán prácticamente siempre por la banda de 2,4 GHz, por ser libre en todo el mundo. El desarrollo de la tecnología se centra en la sencillez y el bajo coste más que otras redes inalámbricas semejantes de la familia WPAN, como por ejemplo Bluetooth. El nodo ZigBee más completo requiere en teoría cerca del 10% del hardware de un nodo Bluetooth o Wi-Fi típico; esta cifra baja al 2% para los nodos más sencillos. No obstante, el tamaño del código en sí es bastante mayor y se acerca al 50% del tamaño del de Bluetooth. Se anuncian dispositivos con hasta 128 kB de almacenamiento.

En 2006 el precio de mercado de un transceptor compatible con ZigBee se acerca al dólar y el precio de un conjunto de radio, procesador y memoria ronda los tres dólares. En comparación, Bluetooth tenía en sus inicios (en 1998, antes de su lanzamiento) un coste previsto de 4-6 dólares en grandes volúmenes. A principios de 2007, el precio de dispositivos de consumo comunes era de unos tres dólares.

La primera versión de la pila suele denominarse ahora ZigBee 2004. La segunda versión y actual a junio de 2006 se denomina ZigBee 2006, y reemplaza la estructura MSG/KVP con una librería de clusters, dejando obsoleta a la anterior versión. ZigBee Alliance ha comenzado a trabajar en la versión de 2007 de la pila para adecuarse a la última versión de la especificación, en concreto centrándose en optimizar funcionalidades de nivel de red (como agregación de datos). También se incluyen algunos perfiles de aplicación nuevos, como lectura automática,automatización de edificios comerciales y automatización de hogares en base al principio de uso de la librería de clusters.

En ocasiones ZigBee 2007 se denomina Pro, pero Pro es en realidad un perfil de pila que define ciertas características sobre la misma.

El nivel de red de ZigBee 2007 no es compatible con el de ZigBee 2004-2006, aunque un nodo RFD puede unirse a una red 2007 y viceversa. No pueden combinarse routers de las versiones antiguas con un coordinador 2007.

Usos

Los protocolos ZigBee están definidos para su uso en aplicaciones embebidas con requerimientos muy bajos de transmisión de datos y consumo energético. Se pretende su uso en aplicaciones de propósito general con características autoorganizativas y bajo costo (redes en malla, en concreto). Puede utilizarse para realizar control industrial, albergar sensores empotrados, recolectar datos médicos, ejercer labores de detección de humo o intrusos o domótica. La red en su conjunto utilizará una cantidad muy pequeña de energía de forma que cada dispositivo individual pueda tener una autonomía de hasta 5 años antes de necesitar un recambio en su sistema de alimentación.

ZigBee vs. Bluetooth

ZigBee es muy similar al Bluetooth pero con algunas diferencias:

- Una red ZigBee puede constar de un máximo de 65535 nodos distribuidos en subredes de 255 nodos, frente a los 8 máximos de una subred (Piconet) Bluetooth.
- Menor consumo eléctrico que el de Bluetooth. En términos exactos, ZigBee tiene un consumo de 30mA transmitiendo y de 3uA en reposo, frente a los 40mA transmitiendo y 0.2mA en reposo que tiene el Bluetooth. Este menor consumo se debe a que el sistema ZigBee se queda la mayor parte del tiempo dormido, mientras que en una comunicación Bluetooth esto no se puede dar, y siempre se está transmitiendo y/o recibiendo.
- Tiene un velocidad de hasta 250 kbps, mientras que en Bluetooth es de hasta 1 Mbps.
- Debido a las velocidades de cada uno, uno es más apropiado que el otro para ciertas cosas. Por ejemplo, mientras que el Bluetooth se usa para aplicaciones como los teléfonos móviles y la informática casera, la velocidad del ZigBee se hace insuficiente para estas tareas, desviándolo a usos tales como la Domótica, los productos dependientes de la batería, los sensores médicos, y en artículos de juguetería, en los cuales la transferencia de datos es menor.
- Existe una versión que integra el sistema de radiofrecuencias característico de Bluetooth junto a una interfaz de transmisión de datos vía infrarrojos desarrollado por IBM mediante un protocolo ADSI y MDSI.

Tipos de dispositivos

Se definen tres tipos distintos de dispositivo ZigBee según su papel en la red:

- **Coordinador ZigBee (ZigBee Coordinator, ZC):** El tipo de dispositivo más completo. Debe existir uno por red. Sus funciones son las de encargarse de controlar la red y los caminos que deben seguir los dispositivos para conectarse entre ellos.

- **Router ZigBee (ZigBee Router, ZR):** Interconecta dispositivos separados en la topología de la red, además de ofrecer un nivel de aplicación para la ejecución de código de usuario.
- **Dispositivo final (ZigBee End Device, ZED):** Posee la funcionalidad necesaria para comunicarse con su nodo padre (el coordinador o un router), pero no puede transmitir información destinada a otros dispositivos. De esta forma, este tipo de nodo puede estar dormido la mayor parte del tiempo, aumentando la vida media de sus baterías. Un ZED tiene requerimientos mínimos de memoria y es por tanto significativamente más barato.

Como ejemplo de aplicación en Domótica, en una habitación de la casa tendríamos diversos Dispositivos Finales (como un interruptor y una lámpara) y una red de interconexión realizada con Routers ZigBee y gobernada por el Coordinador.

Funcionalidad

Basándose en su funcionalidad, puede plantearse una segunda clasificación:

- Dispositivo de funcionalidad completa (FFD): También conocidos como nodo activo. Es capaz de recibir mensajes en formato 802.15.4. Gracias a la memoria adicional y a la capacidad de computar, puede funcionar como Coordinador o Router ZigBee, o puede ser usado en dispositivos de red que actúen de interface con los usuarios.

- Dispositivo de funcionalidad reducida (RFD): También conocido como nodo pasivo. Tiene capacidad y funcionalidad limitadas (especificada en el estándar) con el objetivo de conseguir un bajo coste y una gran simplicidad. Básicamente, son los sensores/actuadores de la red.

- Un nodo ZigBee (tanto activo como pasivo) reduce su consumo gracias a que puede permanecer dormido la mayor parte del tiempo (incluso muchos días seguidos). Cuando se requiere su uso, el nodo ZigBee es capaz de despertar en un tiempo ínfimo, para volverse a dormir cuando deje de ser requerido. Un nodo cualquiera despierta en aproximadamente 15 ms. Además de este tiempo, se muestran otras medidas de tiempo de funciones comunes:

- Nueva enumeración de los nodos esclavo (por parte del coordinador): aproximadamente 30 ms.

- Acceso al canal entre un nodo activo y uno pasivo: aproximadamente 15 ms.

Protocolos

Los protocolos se basan en investigaciones recientes sobre algoritmos de red (ad hoc on-demand distance vector, vector de distancias bajo demanda; neuRFon) para la construcción de redes ad-hoc de baja velocidad. La mayoría de redes grandes están pensadas para formar un cluster de clusters. También puede estructurarse en forma de malla o como un solo cluster. Los perfiles actuales de los protocolos soportan redes que utilicen o no facilidades de balizado.

Las redes sin balizas (aquéllas cuyo grado de balizado es 15) acceden al canal por medio de CSMA/CA. Los routers suelen estar activos todo el tiempo, por lo que requieren una alimentación estable en general. Esto, a cambio, permite redes heterogéneas en las que algunos dispositivos pueden estar transmitiendo todo el tiempo, mientras que otros sólo transmiten ante la presencia de estímulos externos. El ejemplo típico es un interruptor inalámbrico: un nodo en la lámpara puede estar recibiendo continuamente ya que está conectado a la red; por el contrario, un interruptor a pilas estaría dormido hasta que el mecanismo se activa. En una red así la lámpara sería un router o coordinador, y el interruptor un dispositivo final.

Si la red utiliza balizas, los routers las generan periódicamente para confirmar su presencia a otros nodos. Los nodos pueden desactivarse entre las recepciones de balizas reduciendo su ciclo de servicio (duty cycle). Los intervalos de balizado pueden ir desde 5,36 ms a 15,36 ms * 214 = 251,65824 segundos a 250 kbps; de 24 ms a 24 ms * 214 = 393,216 segundos a 40 kbps; y de 48 ms a 48 ms * 214 = 786,432 segundos a 20 kbps. Sin embargo, los periodos largos con ciclos de servicio cortos necesitan que una temporización precisa, lo que puede ir en contra del principio de bajo coste.

En general, los protocolos ZigBee minimizan el tiempo de actividad de la radio para evitar el uso de energía. En las redes con balizas los nodos sólo necesitan estar despiertos mientras se transmiten las balizas (además de cuando se les asigna tiempo para transmitir). Si no hay balizas, el consumo es asimétrico repartido en dispositivos permanentemente activos y otros que sólo no están esporádicamente.

Los dispositivos ZigBee deben respetar el estándar de WPAN de baja tasa de transmisión IEEE 802.15.4-2003. Éste define los niveles más bajos: el nivel físico (PHY) y el control de acceso al medio (MAC, parte del nivel de enlace de datos, DLL). El estándar trabaja sobre las bandas ISM de uso no regulado detalladas más arriba. Se definen hasta 16 canales en el rango de 2,4 GHz, cada uno de ellos con un ancho de banda de 5 MHz. La frecuencia central de cada canal puede calcularse como: FC = (2405 + 5*(k-11)) MHz, con k = 11, 12, ..., 26.

Las radios utilizan un espectro de dispersión de secuencia directa. Se utiliza BPSK en los dos rangos menores de frecuencia, así como un QPSK ortogonal que transmite dos bits por símbolo en la banda de 2,4 GHz. Ésta permite tasas de transmisión en el

aire de hasta 250 kbps, mientras que las bandas inferiores se han ampliado con la última revisión a esta tasa desde los 40 kbps de la primera versión. Los rangos de transmisión oscilan entre los 10 y 75 metros, aunque depende bastante del entorno. La potencia de salida de las radios suele ser de 0 dBm (1 mW).

Si bien en general se utiliza CSMA/CA para evitar colisiones en la transmisión, hay algunas excepciones a su uso: por una parte, las tramas siguen una temporización fija que debe ser respetada; por otra, las confirmaciones de envíos tampoco siguen esta disciplina; por último, si se asignan slots de tiempo garantizados para una transmisión tampoco es posible que exista contención.

Hardware y software

El software se ha diseñado para ejecutarse en procesadores y microcontroladores de bajo coste, con un diseño de radio muy optimizado para lograr bajos costes con altos volúmenes de producción. Utiliza circuitos digitales siempre que es posible y evita los componentes analógicos.

Si bien el hardware es sencillo, el proceso de certificación de un dispositivo conlleva una validación completa de los requerimientos del nivel físico. Esta revisión intensiva tiene múltiples ventajas, ya que todas las radios fabricadas a partir de una misma máscara de semiconductor gozarán de las mismas características de radiofrecuencia. Por otro lado, un nivel físico mal controlado podría perjudicar no sólo al propio dispositivo, sino al consumo de energía de otros dispositivos en la red. Otros estándares pueden compensar ciertos problemas, mientras que ZigBee trabaja en márgenes muy estrechos de consumo y ancho de banda. Por ello, según el 802.15.4, las radios pasan validaciones ISO 17025. La mayoría de fabricantes planea integrar la radio y el microcontrolador en un único chip.

Topologías de red

ZigBee permite tres topologías de red:

- Topología en estrella: el coordinador se sitúa en el centro.
- Topología en árbol: el coordinador será la raíz del árbol.
- Topología de malla: al menos uno de los nodos tendrá más de dos conexiones.

La topología más interesante (y una de las causas por las que parece que puede triunfar ZigBee) es la topología de malla. Ésta permite que si, en un momento dado, un nodo del camino falla y se cae, pueda seguir la comunicación entre todos los demás

nodos debido a que se rehacen todos los caminos. La gestión de los caminos es tarea del coordinador.

Estrategias de conexión de los dispositivos en una red Zigbee

Las redes ZigBee han sido diseñadas para conservar la potencia en los nodos 'esclavos'. De esta forma se consigue el bajo consumo de potencia. La estrategia consiste en que, durante mucho tiempo, un dispositivo "esclavo" está en modo "dormido", de tal forma que solo se "despierta" por una fracción de segundo para confirmar que está "vivo" en la red de dispositivos de la que forma parte. Esta transición del modo "dormido" al modo "despierto" (modo en el que realmente transmite), dura unos 15ms, y la enumeración de "esclavos" dura alrededor de 30ms, como ya se ha comentado anteriormente.

En las redes Zigbee, se pueden usar dos tipos de entornos o sistemas:

- **Con balizas**

Es un mecanismo de control del consumo de potencia en la red. Permite a todos los dispositivos saber cuándo pueden transmitir. En este modelo, los dos caminos de la red tienen un distribuidor que se encarga de controlar el canal y dirigir las transmisiones. Las balizas que dan nombre a este tipo de entorno, se usan para poder sincronizar todos los dispositivos que conforman la red, identificando la red domótica, y describiendo la estructura de la "supertrama". Los intervalos de las balizas son asignados por el coordinador de red y pueden variar desde los 15ms hasta los 4 minutos.

Este modo es más recomendable cuando el coordinador de red trabaja con una batería. Los dispositivos que conforman la red, escuchan a dicho coordinador durante el "balizamiento" (envío de mensajes a todos los dispositivos, entre 0,015 y 252 segundos). Un dispositivo que quiera intervenir, lo primero que tendrá que hacer es registrarse para el coordinador, y es entonces cuando mira si hay mensajes para él. En el caso de que no haya mensajes, este dispositivo vuelve a "dormir", y se despierta de acuerdo a un horario que ha establecido previamente el coordinador. En cuanto el coordinador termina el "balizamiento", vuelve a "dormirse".

- **Sin balizas**

Se usa el acceso múltiple al sistema Zigbee en una red punto a punto cercano. En este tipo, cada dispositivo es autónomo, pudiendo iniciar una conversación, en la cual los otros pueden interferir. A veces, puede ocurrir que el dispositivo destino puede no oír la petición, o que el canal esté ocupado.

Este sistema se usa típicamente en los sistemas de seguridad, en los cuales sus dispositivos (sensores, detectores de movimiento o de rotura de cristales), duermen prácticamente todo el tiempo (el 99,999%). Para que se les tenga en cuenta, estos elementos se "despiertan" de forma regular para anunciar que siguen en la red. Cuando se produce un evento (en nuestro sistema será cuando se detecta algo), el sensor "despierta" instantáneamente y transmite la alarma correspondiente. Es en ese momento cuando el coordinador de red, recibe el mensaje enviado por el sensor, y activa la alarma correspondiente. En este caso, el coordinador de red se alimenta de la red principal durante todo el tiempo.

Futuro

Se espera que los módulos ZigBee sean los transmisores inalámbricos más baratos de la historia, y además producidos de forma masiva. Tendrán un coste aproximado de alrededor de los 6 euros, y dispondrán de una antena integrada, control de frecuencia y una pequeña batería. Ofrecerán una solución tan económica porque la radio se puede fabricar con muchos menos circuitos analógicos de los que se necesitan habitualmente.

2.5.3.2. Infrarrojos

Muy extendido y popular en aparatos electrónicos de pequeño alcance como mandos a distancia o periféricos inalámbricos de ordenador. Se basa en un principio sencillo, modular los datos que transmite en trenes de pulsos de diferente longitud y duración. Es el mismo principio que se usa en el código morse para transmitir letras del alfabeto.

Utiliza generalmente para la emisión diodos LED que emiten en el espectro infrarrojo que resulta invisible para el ojo humano. Para la recepción se usaron en primer lugar fotodiodos u otro tipo de componentes pasivos que fueran capaces de reaccionar ante la luz recibida y que han sido cambiados recientemente por circuitos integrados que incorporan algún componente sensible a la luz como fototransistores o los mismo fotodiodos pero que además incluyen toda la electrónica necesaria para procesar las señales recibidas, simplificando en gran medida los componentes adicionales necesarios para montar un circuito receptor de este tipo de señal.

Su ventaja como hemos visto, es su sencillez y amplia difusión, añadida al hecho de que los componentes necesarios para su uso son realmente económicos.

Su principal desventaja es su alcance, que sin complejos juegos de lentes difícilmente llega a la decena de metros y más importante aún, la necesidad de tener una línea visual directa y despejada de objetos opacos entre emisor y receptor.

Esta problemática hace este método de comunicaciones poco adecuado para su uso con robots móviles, que con mucha facilidad pueden dejar de cumplir los requisitos de visibilidad emisor-receptor comentado. Pese a esto, ha sido utilizado en algunos robots comerciales, como es el caso del predecesor del LEGO Minstorms NXT, el modelo RCX.

2.6. LEGO NXT

De todos es sabido el largo camino que lleva recorrido la firma Lego en lo que se conoce como 'construcciones', es decir, juguetes dirigidos a un público generalmente infantil o adolescente. Esta andadura empezó en el año 1934 en Dinamarca y con el tiempo fue cobrando popularidad entre padres, hijos y sobretodo educadores que veían en estos juguetes una forma muy interesante de divertir y a la vez potenciar la creatividad de sus hijos y alumnos.

La oferta de juguetes va desde los pocos meses de edad hasta la adolescencia e incluso un poco más allá, contando con temáticas urbanas, submarinas e incluso espaciales. Entre todas estas líneas cabe destacar una que desde el principio ha contado ha despertado la curiosidad de determinado sector de público, en el que podían englobarse ingenieros y apasionados de la técnica. Hablamos de la denominada línea *Technic*. La principal característica de esta gama de juguetes y la que sin duda la ha hecho tan famosa, es el peculiar enfoque de construcción que ha concebido Lego con ella.

Este enfoque se aparta del juguete tradicional y experimenta con elementos propios de la tecnología actual, tanto en mecánica como en electrónica. Así, entre las piezas de estos 'kits' se encuentran engranajes, ejes, motores eléctricos, cables y hasta elementos tan concretos como levas y pistones, juntas cardan y amortiguadores.

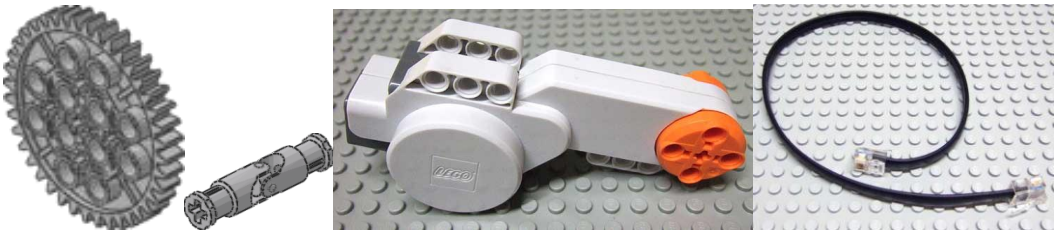


Figura 45: De izquierda. a derecha, engranaje, junta cardan, motor y cable de LEGO

Con estos elementos se emula a la perfección la realidad y se permite construir representaciones a escala de vehículos y sistemas con mecanismos a imagen y semejanza de sus modelos reales, teniendo incluso un funcionamiento análogo o lo más aproximado posible, a como lo hacen dichos dispositivos en la realidad.

Figura 21: Diferentes modelos de la línea LEGO Technic

Sin embargo, los tiempos en que los juguetes tradicionales, y en especial las construcciones, reinaban entre sus potenciales consumidores están tocando a su fin de la mano de la revolución tecnológica. De esta forma han sido prácticamente desbancados por dispositivos con silicio en sus entrañas, como consolas, ordenadores y demás parafernalia electrónica. LEGO por lo tanto, siguiendo el ejemplo de sus clientes, no se ha conformado con ésta aproximación a la tecnología moderna de finales de siglo, en su día revolucionaria, sino que como mandan los tiempos ha decidido ir más allá.

2.6.1. Ladrillo NXT

Tradicionalmente la línea Technic de LEGO ha hecho sus incursiones en el campo de la tecnología electrónica, añadiendo a su oferta de luces, motores, baterías y mandos con hilos que pese al atractivo que ofrecían en si día, han quedado atrás frente a los elementos de última tecnología que a día de hoy copan el mercado de la electrónica de consumo. Por ello, LEGO ha tomado el testigo tecnológico incorporando lo que ha venido a llamarse como '*bricks*' o '*ladrillos*' inteligentes a varias de sus líneas de productos.

Figura 47: De izquierda a derecha versiones RCX y NXT del ladrillo de control

Estos 'ladrillos', proveen de una unidad de control para robots o sistemas electrónicos avanzados que permite implementar escenarios hasta el momento impensables en un juego de construcciones, incluso los de la línea Technic de la propia marca. Su corazón está formado por un microcontrolador o procesador, que en sus últimas versiones ha alcanzado grados de potencia sorprendentes si pensamos en ellos como en un juguete.

Por ello, cada vez más personas se han interesado en estos sistemas, no sólo para la docencia o el ocio, sino incluso para la investigación. Prácticamente todas las universidades a nivel mundial poseen ya kits de LEGO con estos *bricks* inteligentes y se usan en las ramas de la enseñanza y investigación como las relacionadas con la robótica o la mecatrónica.

El modelo usado en la realización de las experiencias que comprende este trabajo es el último disponible en el mercado, el Mindstorms NXT. Este está formado por un kit que incluye: el *brick* NXT, múltiples sensores (luz/color, ultrasonidos, contacto, sonido) motores y cableado eléctrico, y el set de construcción habitual de la línea Technic, formado por miles de piezas de construcción, engranajes, ejes correas y poleas.

A continuación una lista de las características electrónicas del *brick* NXT, extraídas del manual de LEGO:

Procesador principal: Atmel® 32-bit ARM® processor, AT91SAM7S256
- 256 KB FLASH
- 64 KB RAM
- 48 MHz

Co-procesador: Atmel® 8-bit AVR processor, ATmega48
- 4 KB FLASH
- 512 Byte RAM
- 8 MHz

Unidad de comunicaciones Bluetooth CSR BlueCoreTM 4 v2.0 + EDR System
- Soporte de Serial Port Profile (SPP)
- 47 KByte RAM Internos
- 8 MBit FLASH Externos
- 26 MHz

Comunicación vía USB 2.0 (12 Mbit/s)

4 puertos de entrada con interfaz de 6 hilos y soporte para conexiones AD y DA
- 1 puerto de alta velocidad, IEC 61158 Tipo 4/EN 50170 compatible
- todos los puertos de entrada cuentan con soporte del bus **I²C**

3 puertos de salida con interfaz de 6 hilos y soporte para lectura desde encoders

Conectores de 6 hilos estándar industrial, RJ12 con ajuste a derechas

Display grafico LCD de 100 x 64 píxel (blanco y negro)
- Área de visión de 26 x 40.6 mm

Altavoz de salida con resolución de 8-bit
- Soporta tasas de muestreo de 2 a 16 KHz

4 botones de goma para interacción con el usuario

Alimentación: 6 pilas de tipo AA
- Se recomienda usar pilas alcalinas
- También se encuentra disponible una batería de Ion de Litio de 1400 mAH
- Cargador opcional disponible para la batería anterior

2.6.2. Motores

La gama de motores de que consta la línea LEGO Technic es bastante elevada, contando entre sus filas con aproximadamente 13 modelos diferentes. Estos motores van desde los pequeños micromotores hasta la joya de la corona de esta familia que no es otro que el motor incluido en el set del NXT. Este motor posee unas características muy buenas en cuanto a respuesta y rendimiento y cuenta asimismo con un juego de engranajes reductores y un *encoder* rotacional en su interior.

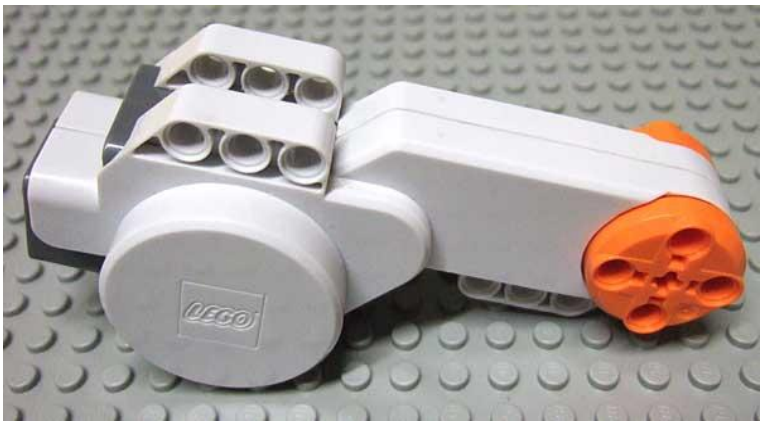


Figura 48: Motor de LEGO incluido en el set NXT

El juego de engranajes reductores garantiza un par de fuerza elevado para el motor, lo que se traduce en la capacidad de realizar un esfuerzo mayor sin la necesidad de elementos externos como era necesario en modelos anteriores, y más importante aún, impidiendo el sobreesfuerzo del motor y la consecuente reducción de su vida útil.

EL *encoder* o tacómetro, por su parte, permite conocer el número de vueltas que ha dado el eje del motor con una precisión de 1 grado. Es decir, si el motor realiza un giro completo el *encoder* verá incrementada su cuenta en 360 unidades. Esto es cierto sea cual sea el sentido de movimiento del motor, incrementándose en un sentido y decrementándose al girar en sentido contrario.

2.6.3. Sensores

En este apartado se realiza un repaso por el conjunto de sensores disponibles para el LEGO Mindstorms NXT, tanto los de marca propia de LEGO como aquellos ensamblados y distribuidos por terceros pero que son plenamente compatibles con la plataforma NXT.

2.6.3.1. Fabricados por LEGO

Son cuatro los sensores incluidos en el kit de LEGO Mindstorms NXT:

- **Bumper:** detecta la presión realizada sobre la punta del sensor. Útil para detectar colisiones, presencia de objetos o como mecanismo final de carrera.
- **Sonido:** consta de un pequeño micrófono capaz de captar sonidos.
- **Luz/Color:** este sensor cumple una doble función, es capaz de detectar la presencia de luz en un entorno y a su vez es capaz de identificar un pequeño rango de colores de los objetos que 've'.
- **Ultrasonidos:** permite detectar la presencia de objetos a una cierta distancia. Útil en las mismas situaciones que el bumper pero sin llegar al contacto.

Figura 4922: Ladrillo NXT con todos sus sensores y motores conectados

2.6.3.2. Fabricados por terceros

Fabricantes como Hi-Technic comercializan sensores compatibles con el NXT, de entre los cuales destacamos los siguientes:

- **Brújula:** permite conocer la orientación del robot respecto al norte magnético.
- **Acelerómetro:** detecta aceleración en tres ejes y giro en uno.
- **Seguidor de infrarrojos:** es capaz de detectar el haz de un emisor en un ángulo de 135º, indicando así mismo la dirección desde la que proviene el haz de luz.

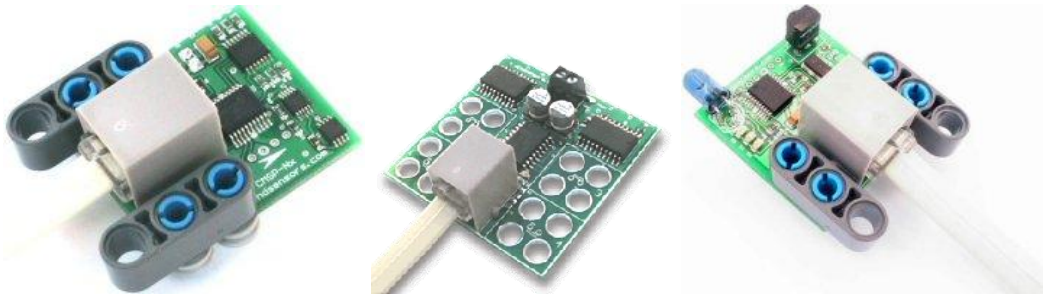


Figura 50: (De derecha a izquierda) Brújula, acelerómetro y seguidor de infrarrojos.

2.6.3.3. Personalizados

Es posible crear sensores propios y personalizados compatibles con el bus I^2C.

2.6.4. Comunicaciones

El LEGO MINDSTORMS NXT ha supuesto una mejora muy importante respecto a su predecesor RCX en el apartado de comunicaciones. Este cambio se debe a que se ha desechado el rudimentario interfaz de comunicaciones infrarrojo y se ha adoptado el nuevo estándar Bluetooth que abre un nuevo mundo de posibilidades.

Los aspectos más relevantes de sistema adoptado se tratan a continuación:

CAPAS DEL SISTEMA DE COMUNICACIÓN:

La siguiente figura muestra las principales capas de la pila de comunicación entre el dispositivo empotrado y el software que se encuentra en el PC. Como se puede comprobar, se puede establecer una comunicación full dúplex entre el PC y el NXT usando o no comunicaciones cableadas. Destaca la capacidad del NXT de usar el protocolo SPP que permite comunicarse vía Bluetooth como si de un puerto serie se tratase.

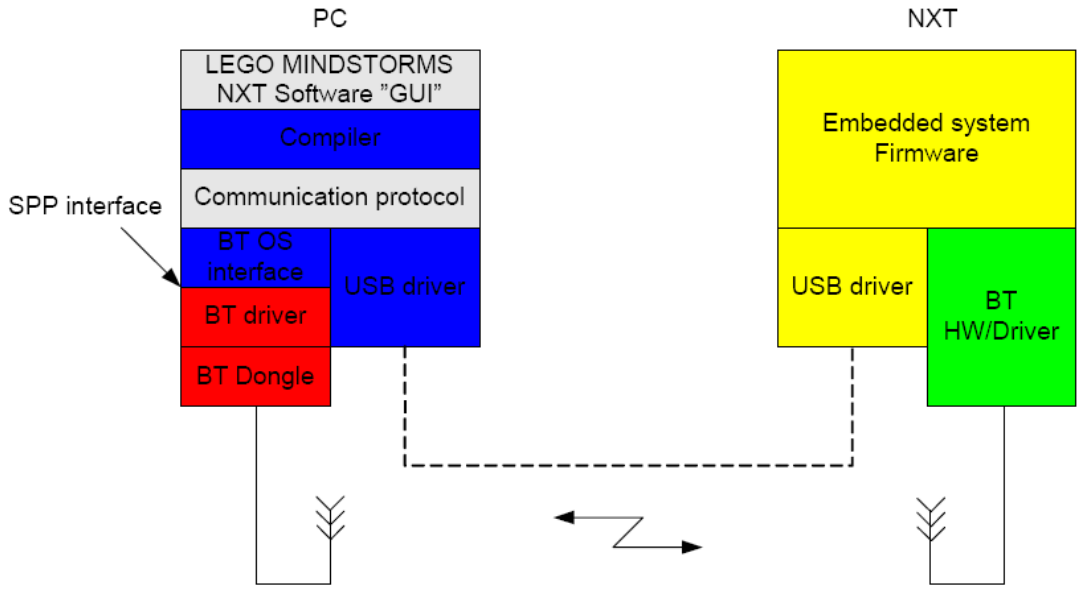


Figura 231: Diagrama de bloques de la comunicación PC-NXT

El firmware que lleva el NXT permite acceder al protocolo de comunicación con el NXT escribiendo o leyendo información en "crudo" o utilizando el protocolo establecido que posteriormente se explicará. La desventaja que se tiene si no se utiliza el protocolo es que es imposible comprobar la información enviada o recibida en el buffer, por lo que se requiere un cierto control a bajo nivel de las tramas enviadas.

Estudios realizados durante el desarrollo han demostrado que dicha comunicación tiene ciertas desventajas cuando se comienza a transmitir ráfagas de información, ya sea porque son demasiado largas y el tiempo entre la recepción de un paquete a otro es demasiado pequeño o por el retardo que se genera cuando se cambia de modo recepción de información a modo de transmisión de información.

Para evitar en la medida de lo posible de la longitud de los paquetes a enviar una solución que se ha implementado es añadir al paquete que se envía el tamaño que ocupa el paquete a enviar al comienzo del paquete.

Para mejorar el retardo que es causado por el cambio de modo de comunicación se ha añadido la posibilidad de enviar comandos directos sin necesitad de reconocimiento, es decir, no se asegura que lo que se envíe al NXT sea procesado correctamente a causa de cualquier tipo de error. Con eso se evita el cambio de modo de transmisión y los 30 ms que antes afectaban, ahora no se producen.

El formato de los paquetes que se envían en ambas direcciones tiene la forma que muestra la figura.

Length, LSB	Length, MSB	Command Type	Command	Byte 5	Byte 6	Etc.

Figura 242: Formato de trama en NXT

Se pueden distinguir, como se observa en la figura, cuatro partes que proporcionan información cuando una trama llega a su destino:

- **Bytes 0 y 1:** Indican el tamaño que tiene el paquete enviado, en dicho tamaño no se contemplan esos dos bytes de información.
- **Byte3:** Tipo de comando. Existen dos tipos de comandos, los comandos directos y los comandos de sistema. Estos a su vez pueden ser con contestación de estado o sin contestación

0x00 → Comando directo con respuesta.
0x80 → Comando directo sin respuesta.
0x01 → Comando de sistema con respuesta.
0x81 → Comando de sistema sin respuesta.
0x02 → Indica que el paquete recibido es la respuesta a otro paquete.

- **Byte 4:** Comando específico. Dentro de los dos tipos posibles hay comandos que tienen un código específico para realizar diferentes funciones dentro del ladrillo NXT.
- **Bytes 5 a N:** Se utiliza para información adicional del comando específico, cada comando tiene un formato que difiere del resto.

En lo que concierne a los paquetes de respuesta el formato que siguen es el que se muestra a continuación:

- **Byte 2** → Siempre vale lo mismo y es 0x02 para indicar respuesta.
- **Byte 3** → Estado de la respuesta. Este valor puede variar según estos valores.

0x00 → Éxito
0x81 → Sin manejadores
0x82→ Sin espacio
0x83→ No hay más ficheros
0x84→ Se esperaba final de fichero
0x85→ Final de fichero
0x86→ No es un fichero lineal
0x87→ Fichero no encontrado
0x88→ Todos los manejadores están cerrados
0x89→ Sin espacio lineal
0x8A → Error desconocido
0x8B → El fichero está ocupado
0x8C → No hay buffers de escritura

0x8D → No se puede añadir
0x8E → El fichero está lleno
0x8F → El fichero existe
0x90 → Módulo no encontrado
0x91 → Fuera del límite
0x92 → Nombre incorrecto de fichero
0x93 → Manejador ilegal

COMANDOS DENTRO DEL PROTOCOLO DE COMUNICACIÓN

En el siguiente apartado se describirá la arquitectura del protocolo de comunicación del NXT. Esta capa está situada debajo de los puertos de comunicación tanto del de USB como del de Bluetooth ya que será la que maneje la información que será intermediada dentro de las capas de comunicación USB y Bluetooth.

En cada una de las capas que forman la pila del protocolo se comprueban los errores y la configuración de los paquetes a enviar tanto para USB como para Bluetooth.

Dejando a un lado las características físicas del protocolo se va a abordar los posibles ficheros que la nueva comunicación del NXT es capaz de manejar. Se pueden agrupar los tipos de ficheros en:

- **Descargas**
  - De firmware.
  - De programas definidos por el usuario.
  - De programas generados por una aplicación externa (pe. RobotC)
  - Programas de prueba (ficheros pre-programados).
  - Sonidos y gráficos.
- **Subidas**
  - De programas.
  - De gráficos y sonidos.
  - De información de las salidas de los programas (logs).
- **Borrados de información en el brick**
  - Sonidos, gráficos.
  - De programas definidos por el usuario.
  - Programas de prueba (ficheros pre-programados).
  - De programas generados por una aplicación externa (pe. RobotC).
  - Ficheros de salida de los programas (logs).
- **Comandos de comunicación con el NXT**
  - Obtener la lista de ficheros dentro del NXT.
  - Comandos dirigidos a la máquina virtual (VM).

- Mensajes de comunicación con los buzones (buffers donde se almacena la información de salida y entrada de NXT).
- Datos directos desde el puerto de alta velocidad.

2.6.5. Software

El éxito de la programación de los bricks del NXT ha dado lugar a la implementación de diferentes lenguajes para desarrollar aplicaciones que controlen los sensores, los motores, las comunicaciones… en definitiva, a un gran abanico de posibilidades según sea el lenguaje preferido para el programador. Tanto es así que al poder acceder al código del firmware al ser de libre distribución se puede adaptar a diferentes Sistemas Operativos. De hecho, existe una versión para Linux que posibilita la programación de estos robots móviles sin tener que abandonar su querido entorno de trabajo habitual.

En los siguientes puntos se van a citar una serie de lenguajes para compararlos entre sí y ver para qué funciones un lenguaje ofrece más facilidades y rapidez que otros.

2.6.5.1. LeJOS

LeJOS es un firmware disponible libremente para instalar y reemplazar el original de los ladrillos de Lego. Existen dos variantes, LeJOS NXJ para el ladrillo NXT y LeJOS RCX para el ladrillo RCX. Este software incluye la máquina virtual Java, por lo que permite a los robots de Lego Mindstorms que se programen en el lenguaje Java y además que se pueda utilizar tanto en Windows como en Linux junto a cualquier software compilador de Java como Eclipse o NetBeans.

Aunque es un software prácticamente en desarrollo, poco a poco se le han ido añadiendo interesantes funcionalidades como navegación, analizadores de visión y otros servicios.

2.6.5.2. LEGO NXT-G

Al contrario que con LeJOS, Lego NXT-G está diseñado para trabajar con el firmware original de Lego. Este software está basado en Labview, un lenguaje por bloques que es intuitivo, rápido y sencillo de programar que ha sido producido en colaboración con National Instruments.

Por defecto este lenguaje lleva programadas rutinas que hacen que aún sea más rápida la programación. El problema que tiene este lenguaje es que está destinado

para alumnos de enseñanza obligatoria con lo cual el aprovechamiento que puede proporcionar esta herramienta no es muy elevado.

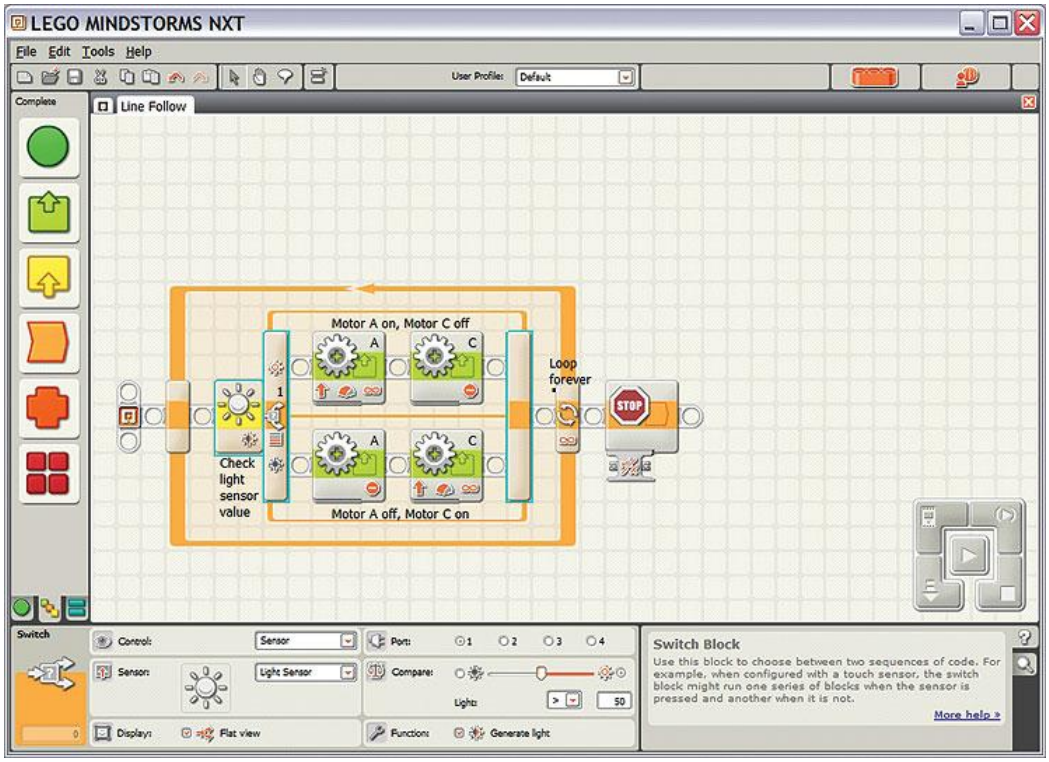


Figura 53: Intefaz Lego NXT-G

2.6.5.3. RobotC

Es un lenguaje muy parecido a C, de hecho parece un micro C que hace uso de un firmware más rápido que el original. Incluye una interfaz de programación típica de texto, con ayudas de rellenado, sintaxis en color, una barra lateral donde se describen las funciones que tiene implementadas RobotC para ayudar a controlar los sensores, etc. y además un depurador para facilitar la labor a los programadores. Dentro de la aplicación se pueden encontrar numerosos ejemplos escritos en dicho lenguaje para probarlos directamente con el robot para comprobar su funcionamiento.

Para mostrar cómo resulta el entorno de programación se adjuntan diferentes capturas donde se observan diferentes partes, como puede ser la parte de las funciones y variables que tiene intrínsecas RobotC, la parte de edición de texto con su resaltado de texto y la ventana de depuración donde se muestran los errores (Figura 9).

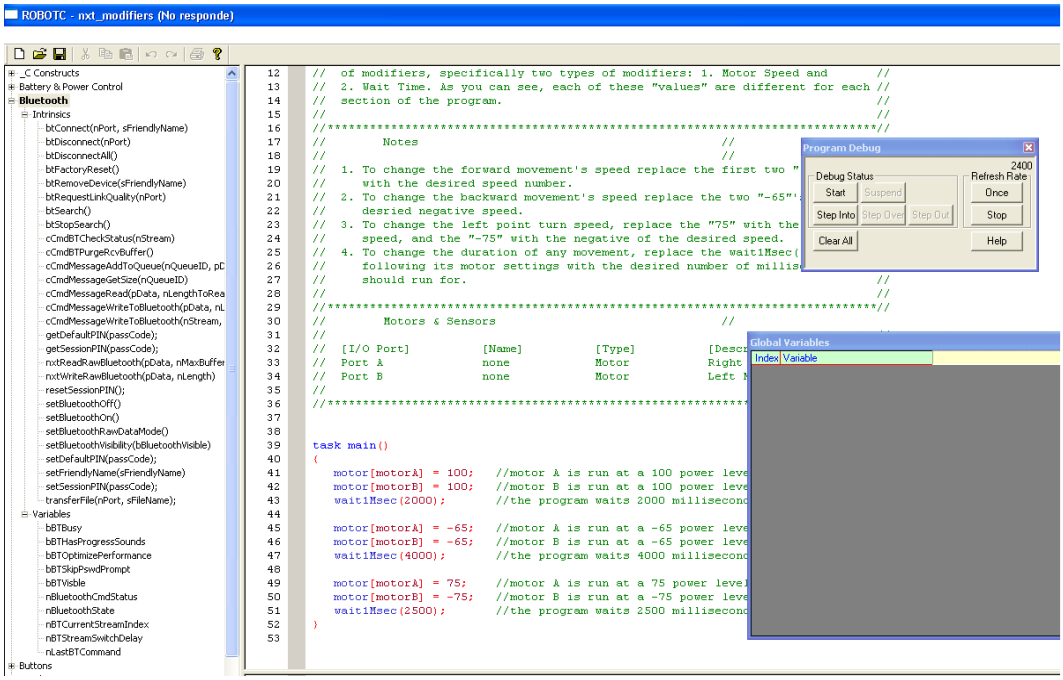


Figura 5425: IDE de programación RobotC

2.6.5.4. Otras alternativas

•**BricxCC:** Bricx Command Center es un conocido IDE que soporta programación del RCX con NQC, C, C++, Pascal, Forth, y Java utilizando brickOS, pbForth y LeJOS. Con BricxCC se pueden desarrollar programas en NBC y NXC. Tanto NBC como NXC utilizan el firmware estándar del NXT. Este software está disponible en código abierto. John Hansen ofrece diversas utilidades, que poco a poco va incluyendo en BricxCC.

•**NBC (Next Byte Codes)**: NBC es un lenguaje dirigido a programadores con una síntesis de lenguaje ensamblador. Se puede utilizar como editor BricxCC. Sorosy.com ofrece un depurador para la programación con NBC.

•**NXC:** NXC es un lenguaje de alto nivel similar a C. Utiliza el firmware original de LEGO y está disponible para Windows, Mac OSX y Linux (ia32). Ha sido desarrollado por John Hansen. Hay disponible una guía del programador y un tutorial en ingles (http://bricxcc.sourceforge.net/nbc) y se puede utilizar como editor Bricxcc.

•**pbLua:** Es un lenguaje textual heredero del pbForth desarrollado para el RCX. Está basado en Lua y tiene como principales características estar escrito en C con mínimos recursos de ejecución, se puede compilar en el NXT, es pequeño, fácil de leer y escribir y hay documentación disponible en la red.

•**NXT#:** Dentro de los lenguajes que interesan este es uno de ellos. Se trata de un lenguaje escrito en C# de Microsoft para la plataforma .NET que tiene

implementadas una serie de métodos que nos permiten controlar parte de la funcionalidad del NXT por una aplicación escrita en C# o Visual Basic .NET través de Bluetooth. Solamente se puede utilizar bajo Windows.

La lista de lenguajes que se ha expuesto es algo más extensa pero no era la finalidad de este apartado comentar todas las posibilidades. Sin embargo se va a proceder a agrupar todas las características comunes a todos los lenguajes para hacer una comparativa mucho más visual y rápida. En las siguientes tablas no se recogen todas las características que tiene cada uno de los lenguajes pero puede ser suficiente para hacerse una idea sobre que lenguaje que mejor se adapte a sus necesidades.

Características	NXT-G Retail	NXT-G Educational	RoboLab 2.9	NBC	NXC	Robot C	NI LabVIEW Toolkit	leJOS NXJ	pbLua	LEJOS OSEK
Tipo de lenguaje	Gráfico	Gráfico	Gráfico	Ensamblador	Como C	C	Gráfico	Java	Lua	ANSI C
Firmware	Estándar	Standard	Estándar(#1)	Estándar	Estándar	Estándar (#1)	Estándar	Modificado	Modificado	Modificado
IDE (¿incluido?)	Si	Si	Si	Si	Si	Si	No (#6)	plugins para Eclipse y NetBeans	No (#7)	Eclipse CDT(GCC+ ATMEL SAM-BA)
Windows	Si	Si	Si	Si	Si	Si	Si	Si	Si (#7)	Si
Mac OSX	Si	Si	Si	Si	Si	Aún no	Si	Si	Si (#7)	No
Linux	No	No	No	Si	Si	No	No	Si	Si (#7)	No (puede)
Eventos	No	No	Si	No	No	Si	No	Eventos estándar de java		Si (OSEK RTOS)
Multitarea	Si	Si	Si	Si	Si	Si	Si	Si		Si (OSEK RTOS)
Bluetooth Brick hacia PC	Si	Si	No	Si	Si	Si	Si	Si	Si	Si
Bluetooth Brick hacia Brick	Si	Si	No	Si	Si	Si	Si	Si	Si	Aún no
Bluetooth Brick hacia otro dispositivo	No	No	No	No	No	Si	No	Si	Si	No
I2C Support	(#5)	(#5)	Si	Si	Si	Si	Si	Si	Si	Si (EEUU sólo)
File System	Si	Si	Si	Si	Si	Si	Si	Si	Aún no	Sin planear
Floating Point	No	No	Si	No	No	Si	¿No?	Si	(#8)	Si
Datalog	No	No	Si	No	No	Si	¿No?	Si	No	Aún no

Figura 55: Comparativa de diferentes lenguajes de desarrollo para NXT

Software	Tipo de Lenguaje	Tipo de Control	Requiere Firmware del NXT	Tipo de Conexión	Fuente de Conexión	Windows	Mac OSX	Linux	Lectura de Sensores	Websites
LEGO NXT Mobile Application	Simple RC	Control Remoto	Estándar (#2)	Bluetooth	Teléfono o PDA	-	-	-	No	LEGO
FunkNXT	Simple RC	Control Remoto	Estándar	Bluetooth	Teléfono	-	-	-	Si	FunkNXT
BT RC	NXT-G	NXT a NXT remotamente	Programa ejecutándose en el NXT	Bluetooth	Otro NXT	-	-	-	Programable por el usuario	BTRC
Simple BT Remote	Simple RC	Control Remoto	Estándar	Bluetooth	Escritorio	Si	No	No	Si	Simple Windows RC
RobotC	Simple RC	Control Remoto	Estándar (#1)	USB/BT	Escritorio	Si	Aún no	No	Si	Robot C Web Site
BricxCC	Simple RC	Control Remoto	Estándar	USB/BT	Escritorio	Si		No	Si	BricxCC Web Site
OnBrick PDA	Gráfico	RC Programable	Estándar	Bluetooth	PDA	-	-	-	Si	OnBrick
OnBrick PC	Gráfico	RC Programable	Estándar	Bluetooth	Escritorio	Si	No	No	Si	OnBrick
NXT Director	Simple RC	Control Remoto Modificable	Estándar	Bluetooth	Palm PDA	-	-	-	¿No?	Director
RoboDNA	Simple RC	Control Remoto	Estándar	Bluetooth	Escritorio	Si			Si	RoboDNA
MS Robotics Studio	.NET	Programa de usuario ejecutándose en el PC	Estándar	Bluetooth	Escritorio	Si			Si	(version no commercial gratis) Download site o Microsoft Site
NI LabVIEW Toolkit	Gráfico (LabVIEW G)	Programa de usuario ejecutándose en el PC	Estándar	USB/BT	Escritorio	Si	Si		Si	LabVIEW toolkit Site
RoboLab	Gráfico	Programa de usuario ejecutándose en el PC	Estándar	USB	Escritorio	Si	Si		Si	Robolab
iCommand	Java	Programa de usuario ejecutándose en el PC	Estándar	Bluetooth	Escritorio o PDA	Si		Si	Si	iCommand
LEGO::NXT	Perl	Programa de usuario ejecutándose en el PC	Estándar	USB/BT	Escritorio	Si	Si	Si	Si	Perl
nxt-Ruby	Ruby	Programa de usuario ejecutándose en el PC	Estándar	Bluetooth	Escritorio	Si	Si	Si	Si	Ruby
NXT#	C#	Programa de usuario ejecutándose en el PC	Estándar	Bluetooth	Escritorio	Si			¿Si?	NXT#
Mindsqualls	C#	Programa de usuario ejecutándose en el PC	Estándar	Bluetooth	Escritorio	Si			Si	Mindsqualls
NXT Python	Python	Programa de usuario ejecutándose en el PC	Estándar	USB/BT	Escritorio	Si			¿Si?	Python
My Robot Me	¿Gráfico?	Programa de usuario ejecutándose en el PC	Estándar	USB/BT	Escritorio	Si	No	No	Si	Robot Me

Notas

(1) RobotC usa firmware estándar que proviene del software de LEGO.
(2) Aplicaciones de LEGO para móviles pueden enviar mensajes a los programas que están ejecución en un NXT.

Figura 56: Comparativa de las distintas funcionalidades.

2.7. Entorno de desarrollo de software

Para el trabajo llevado a cabo se estudiaron las posibles opciones disponibles para desarrollar el software. Las dos principales se detallan a continuación:

2.7.1. Eclipse

Conocido por cualquier programador de lenguaje Java, Eclipse fue desarrollado originalmente por IBM como el sucesor de su familia de herramientas para VisualAge. Hoy en día, lo desarrolla la Fundación Eclipse, una organización independiente sin ánimo de lucro que fomenta una comunidad de código abierto y un conjunto de productos complementarios, capacidades y servicios. A continuación se muestra una tabla resumen de la situación actual del software.

Estado actual de Eclipse	
Líneas de código fuente	2,063,083
Esfuerzo estimado de desarrollo(persona-año / persona-mes)	604.33 / 7,251.93
Estimación de tiempo (años-meses)	6.11 / 73.27
Estimación del nº de desarrolladores en paralelo	98.98
Estimación de coste	$ 81,636,459

Figura 57: Estado actual de Eclipse.

Un punto muy importante a notar son los diversos lenguajes de programación utilizados en el desarrollo del proyecto, de acuerdo al análisis realizado usando SLOCCount, el lenguaje más utilizado es Java, seguido de ANSI C.

Lenguajes de programación utilizados en Eclipse 3.2.1		
Lenguaje	**Líneas de código**	**%**
Java	1,911,693	92.66%
ANSI C	133,263	6.46%
C++	10,082	0.49%
JSP	3,613	0.18%
sh	2,066	0.10%
perl	1,468	0.07%
php	896	0.04%
sed	2	0.00%

Figura 58: Lenguajes utilizados con Eclipse.

Actualmente Eclipse continúa en desarrollo y su última versión es Galileo, que corresponde a la versión 3.5 de Eclipse, salió el 24 de junio del 2009.

2.7.2. NetBeans

Una buena alternativa a Eclipse es la plataforma NetBeans. Permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados *módulos*. Un módulo es un archivo Java que contiene clases de java escritas para interactuar con las APIs de NetBeans y un archivo especial (manifest file) que lo identifica como módulo. Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos módulos. Debido a que los módulos pueden ser desarrollados independientemente, las aplicaciones basadas en la

plataforma NetBeans pueden ser extendidas fácilmente por otros desarrolladores de software.

NetBeans también es un proyecto de código abierto de gran éxito con una gran base de usuarios, una comunidad en constante crecimiento, y con cerca de 100 socios en todo el mundo. Sun MicroSystems fundó el proyecto de código abierto NetBeans en junio de 2000 y continúa siendo el patrocinador principal de los proyectos.

Según 'SLOCCount' de David A. Wheeler's, el estado actual de NetBeans es el siguiente:

Estado actual de NetBeans IDE 6.1

Líneas De Código Fuente	1.990.915
Esfuerzo estimado de desarrollo (persona-año / persona-mes)	582,15 / 6.985,81
Estimación de tiempo (años-meses)	6,02 / 72,23
Estimación del nº de desarrolladores en paralelo	96,71
Estimación de coste	$ 78.640.629 (Salario promedio $56,286/año con unos costos extras de 2,40)

Figura 59: Estado actual de NetBeans IDE 6.1

En lo referente a los lenguajes utilizados con NetBeans, Java obtiene un porcentaje de uso del 99,19% lo cual demuestra que es una de las opciones más demandadas por los programadores a la hora de elegir entornos de programación en Java.

3.- Desarrollo práctico

Una vez expuestas las bases teóricas necesarias para el desarrollo de nuestros objetivos, se continúa con la descripción de la parte práctica del proyecto.

3.1. PRÉAMBULOS Y CONFIGURACIONES.

3.1.1. Montaje del robot NXT

Siguiendo el guión de las finalidades de este proyecto, el montaje del robot NXT no puede ser de cualquier manera. Se necesita que cumpla unas necesidades de movilidad, de tracción y de dirección. De las opciones planteadas en la parte teórica, se opta por la forma de triciclo: dos ruedas motrices que controlan tracción y dirección al mismo tiempo con un mecanismo diferencial y una rueda trasera más pequeña y libre.

3.1.2. Java Developement Kit.

Por supuesto si se va a trabajar con el lenguaje Java se necesita instalar la Máquina Virtual Java en el ordenador. Desde su página web http://www.java.com/en/download/manual.jsp se puede descargar la última versión e instalarla. Hay que asegurarse de que se instala al menos la versión 5 ya que es la mínima necesaria para que funcione el entorno de trabajo seleccionado.

3.1.3. Instalación del driver USB de Lego

Para que el ordenador reconozca el robot, se debe descargar desde http://mindstorms.lego.com/en-us/support/files/default.aspx el driver de Lego Mindstorms NXT e instalarlo. Ahora cuando se conecte el robot al ordenador por USB y se encienda, aparecerá entre los dispositivos del sistema.

3.1.4. Instalación de LeJOS NXJ.

Dado que LeJOS NXJ cuenta con su propio firmware, es necesario reemplazar el original que tienen los robots. Para ello se debe descargar el software desde: http://LeJOS.sourceforge.net/nxj-downloads.php e instalarlo en el ordenador.

El software incluye una biblioteca de clases de Java (Classes.jar) que implementan la LEJOS NXJ Application Programming Interface (API), herramientas de PC para actualizar el firmware del robot, cargar los programas, depurarlos, un API de PC para escribir programas de PC que se comunican con los programas de LEJOS NXJ a través de Bluetooth o USB y otras muchas funciones y programas de ejemplo.

Se conecta el robot al ordenador por USB y se utiliza la aplicación nxjflash para actualizar el firmware a la versión de LeJOS NXJ, y con esto los robots quedan preparados.

Para realizar las aplicaciones, pruebas y rutinas, se ha utilizado el entorno de trabajo de ECLIPSE, por su popularidad entre programadores en Java y por la familiaridad personal con la que se contaba anteriormente.

3.1.5. Configuración del entorno de desarrollo (ECLIPSE)

Desde http://www.eclipse.org/downloads/ se descarga la última versión de ECLIPSE IDE para desarrolladores Java y hay que descomprimirla en el disco duro.

Aunque LeJOS NXJ proporciona una aplicación para descargar los programas en el NXT, el hecho de compilar con ECLIPSE y luego usar otra aplicación para descargar los datos al NXT resulta algo engorroso. Por tanto, ya que se va a trabajar en ECLIPSE, se configurará para tener la opción de descargar los programas directamente en el ladrillo, sin necesidad de la aplicación de LeJOS. Para esto hay que configurarlo de la siguiente manera:

Con un proyecto abierto, se hace clic en “Run” -> “External Tools” -> “External Tools Configurations…”

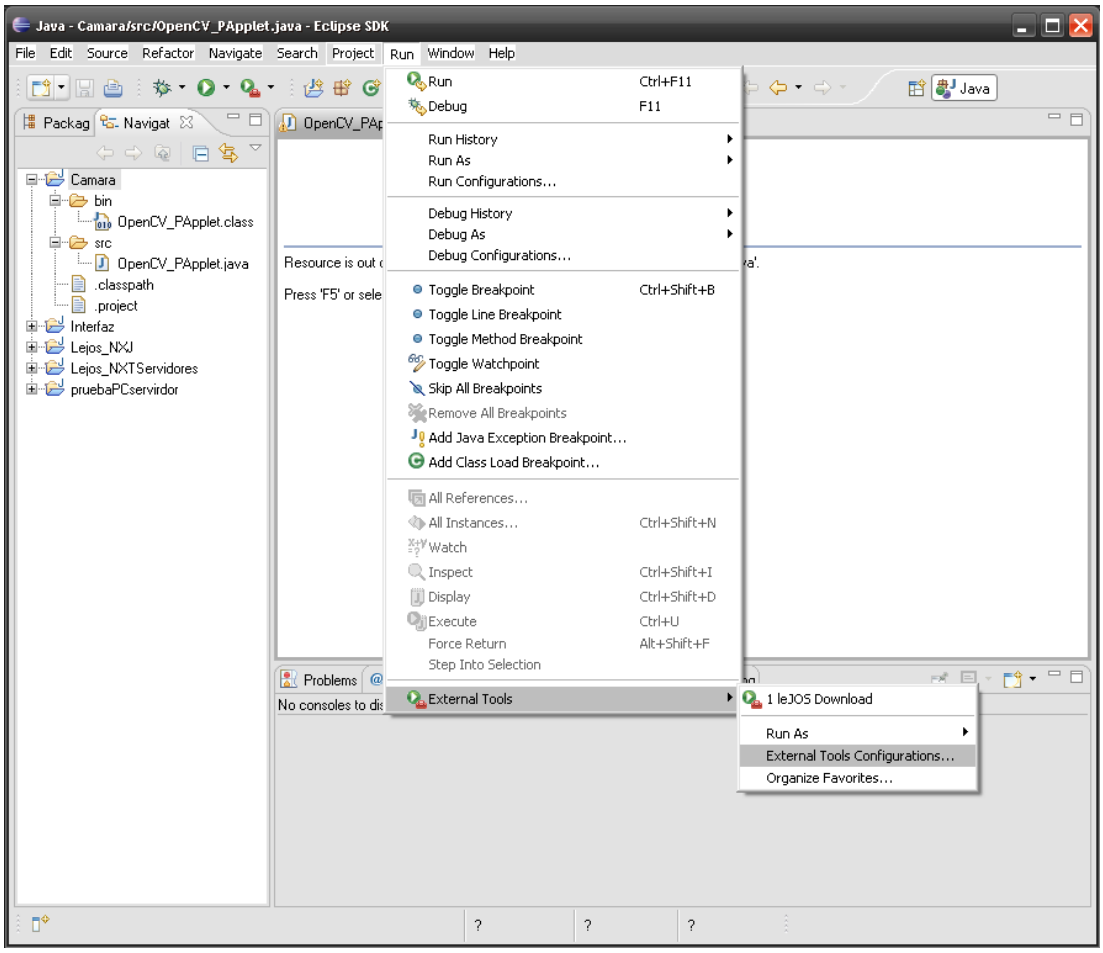


Figura 60: Configuración Eclipse

Se lanza una nueva ventana donde hay que seleccionar “Program” en la parte de la izquierda y el icono de “New launch configuration”. Como nombre se le puede dar “LeJOS Download” por ejemplo y en la pestaña “Main” hay que rellenar el campo “Location” con la ruta donde se encuentra el ejecutable “LeJOSdl.bat” proporcionado por LeJOS. (Deberá estar en %ruta_de_LeJOS_NXT% /bin). Este ejecutable es el que permite descargar directamente al NXT, lo que se está haciendo es vincular el ECLIPSE a esa aplicación para poder hacerlo directamente sin salir del entorno. Siguiendo con la configuración en el campo “Working Directory” hay que introducir “${project_loc}\bin” (sin comillas) y en el campo “Arguments” introducir “${java_type_name}” (sin comillas también). Y aplicar.

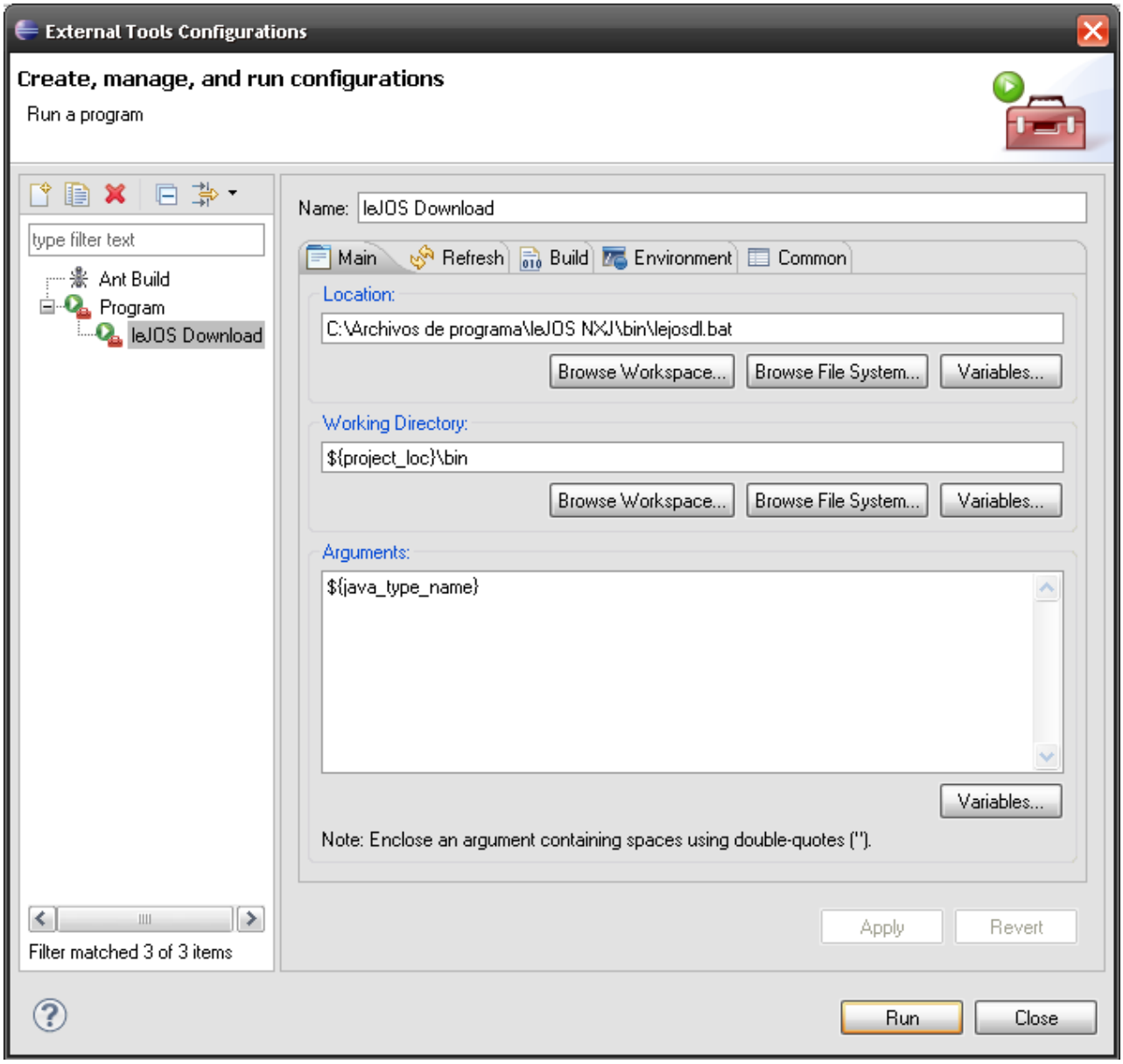


Figura 61: Configuración Eclipse 2.

Cuando ya se ha creado la configuración, se puede poner un acceso directo haciendo clic en el icono de “Run”->”Organize Favorites…”

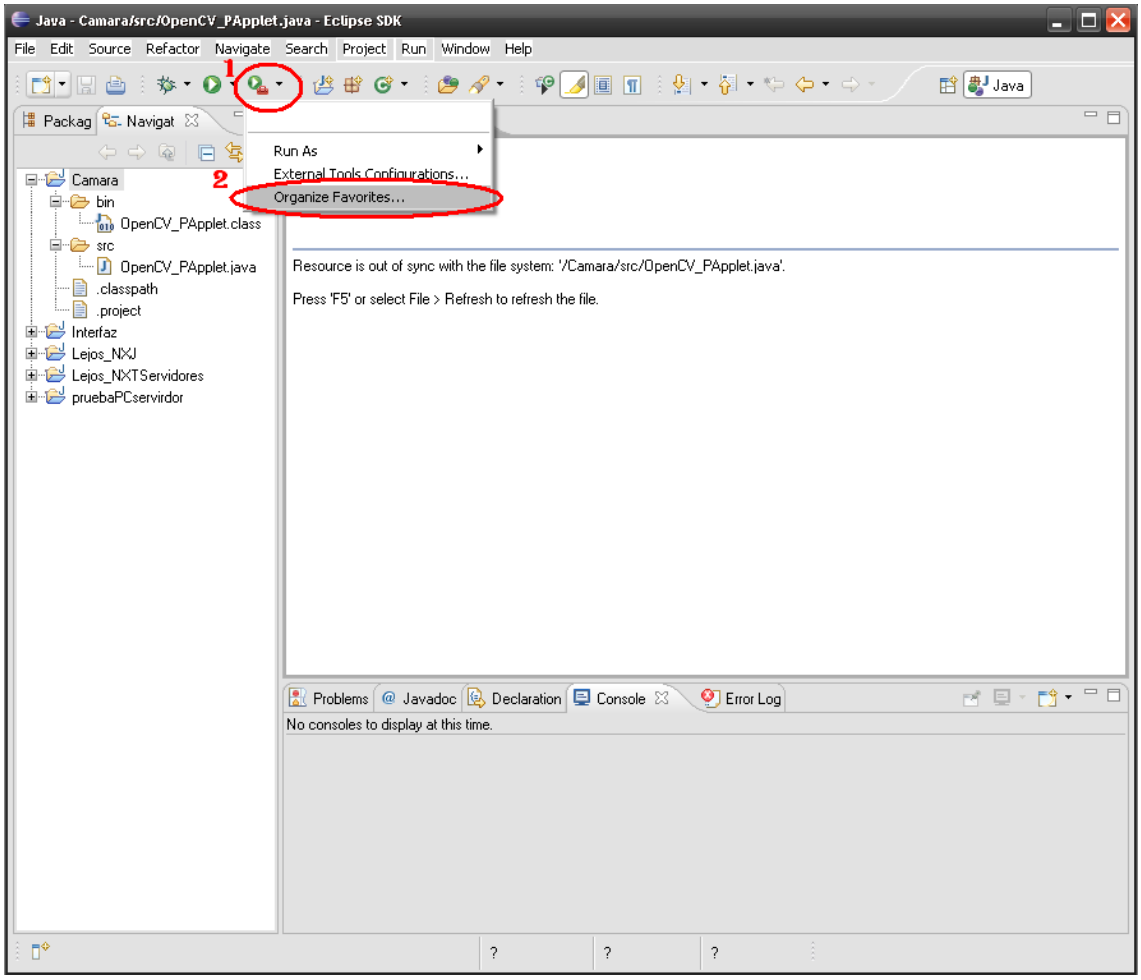


Figura 62: Configuración Eclipse 3.

Dar clic en “Add…” en la ventana nueva que aparece y luego seleccionar la opción de “LeJOS Download” (o el nombre que se le haya querido dar a la configuración definida anteriormente). Presionar “Ok” y luego “Ok” de nuevo.

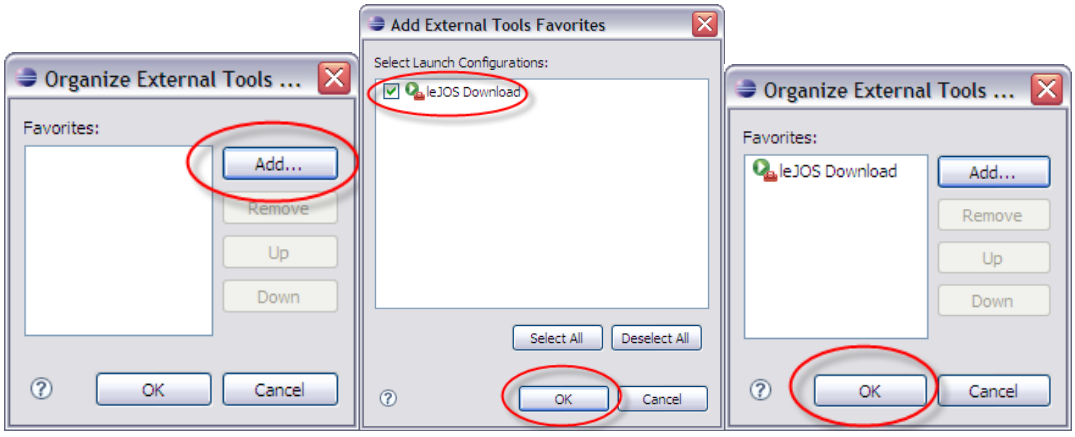


Figura 63: Configuración Eclipse 4.

De este modo con tan sólo hacer clic en el botón “Run” que se ha configurado, Eclipse compilará y en caso de que no dé errores descargará el programa en el NXT mediante USB.

3.2. Estudio de comunicaciones mediante Bluetooth

Se propone descubrir y analizar la comunicación vía Bluetooth de los robots NXT mediante LeJOS NXJ. El buen funcionamiento de las aplicaciones desarrolladas depende directamente de una buena comunicación entre dispositivos, por lo tanto es necesario conocer la eficiencia, la relación del rendimiento de la comunicación en función de los dispositivos conectados, sus limitaciones y la estabilidad de la conexión ante excepciones.

3.2.1. Instalación y configuración del dispositivo Bluetooth

Para poder comunicarse por Bluetooth con el NXT, habrá que instalar un dispositivo que cumpla dicha función. Algunos ordenadores de hoy en día ya incorporan un dispositivo Bluetooth integrado pero no todos son compatibles con los NXT, es recomendable instalar el original de Lego para evitar estos problemas.

Figura 64: Bluetooth Lego

Sin necesidad de drivers, este Bluetooth se instala automáticamente al conectarlo a un puerto USB y trae su propia GUI para explorar dispositivos y conectarse a ellos. Se controlará mediante la librería “bluecove” proporcionada por LeJOS NXJ pero antes es imprescindible que se realice el “enlace” entre los dispositivos que vayamos a utilizar. Mediante el software del dispositivo Bluetooth se debe incluir entre los dispositivos conocidos todos aquellos que se vayan a utilizar. De lo contrario cuando se utilice la librería para acceder a los dispositivos encontrados, no se mostrarán los que no hayan sido enlazados previamente.

3.2.2. Configuración de librerías

Para un correcto funcionamiento y un total acceso a todas las funciones que ofrece LeJOS NXJ, en el Java Build Path del proyecto destinado a ejecutarse en el NXT, habrá que incluir las librerias: classes.jar, jtools.jar, pccomm.jar y pctools.jar proporcionadas por LeJOS. Para el proyecto a ejecutar en el ordenador, bastará con incluir la "bluecove.jar" para controlar el Bluetooth y la "pccomm.jar" que gestiona las comunicaciones.

3.2.3. Tiempos de conexión entre dispositivos.

Se ha comprobado que es mucho más rápido conectarse a un dispositivo conociendo su Nombre y su dirección MAC, que realizando una búsqueda de los dispositivos que se encuentran al alcance. Para el caso que nos ocupa, dado que el dispositivo Bluetooth debe estar enlazado con los robots previamente, siempre se realiza la conexión mediante su Nombre y su MAC.

Aunque no es posible conectar por Bluetooth más de 7 dispositivos a un mismo dispositivo maestro, las posibilidades de configuración de una Scatternet o red dispersa son infinitas.

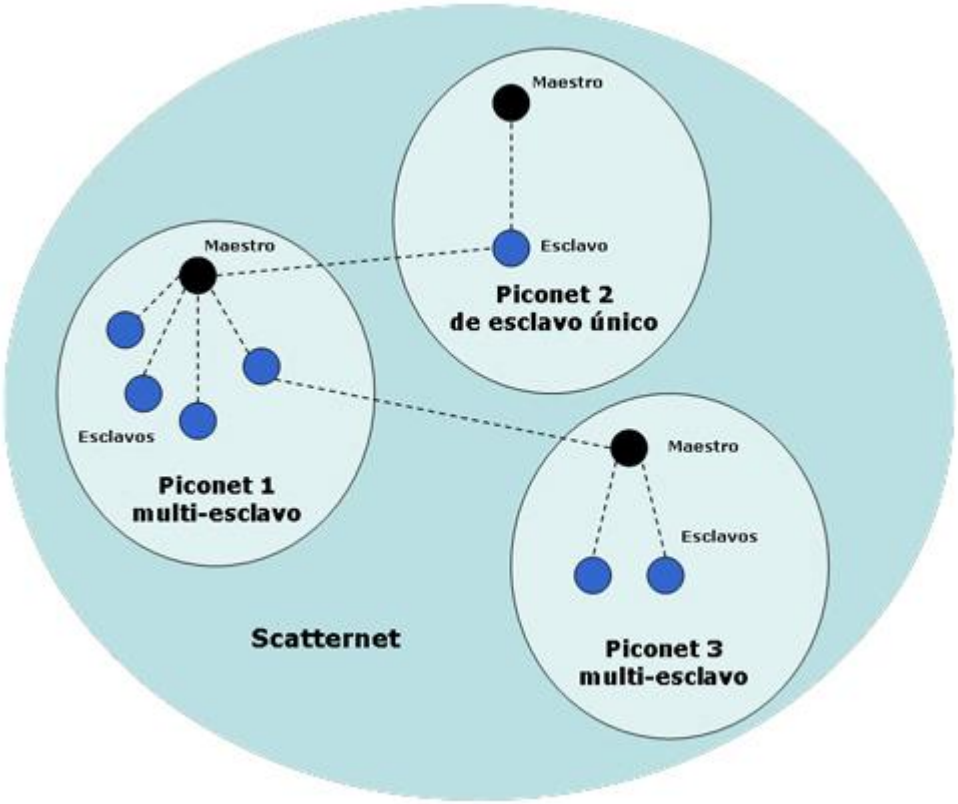


Figura 65: Scatternet.

Según varios programas de prueba que se han realizado, la comunicación responde satisfactoriamente con distintas configuraciones y el tiempo de transferencia de datos entre dispositivos no supera el medio segundo en el peor de los casos.

3.2.4. Estudio de los posibles tipos de datos a transmitir.

Cuando se establece una conexión entre dispositivos mediante Bluetooth, por cada dispositivo es necesario crear dos canales, uno de entrada de datos y otro de salida de datos. Pero estos canales no permiten transmitir cualquier tipo de datos. No es posible enviar archivos completos o variables como matrices o vectores de enteros, por eso para el envío simultáneo de variables más complejas que simples enteros, caracteres o doubles, se ha utilizado el envío de Strings (writeBytes(String string)).

Se ha estudiado cómo influye la dimensión del String en el tiempo de transferencia, la tabla se adjunta a continuación:

Dimensión (caracteres)	Tiempo de respuesta
100	359
200	438
400	453
500	453
550	531
600	516
1000	625

Figura 66. Tabla de envíos dimensión-respuesta.

Como se puede observar, es más rápido enviar un String con una dimensión de 1000 caracteres, que hacer dos envíos de 500. Por lo que la opción más rápida parece ser limitar el número envíos al mínimo y compactarlos en envíos de String de mayor dimensión siempre y cuando sea posible.

3.3. Logger

Antes de comenzar con la gestión de trayectorias, es interesante investigar la manera de crear un pequeño logger, que ofrezca datos precisos de las posiciones del NXT y así no guiarse tan sólo de un análisis visual de los resultados de las trayectorias.

La primera idea desarrollada consiste en incluir un hilo “Rastreador” en cada robot que cada cierto período de tiempo captura las coordenadas del robot y las envía al PC. En el PC será necesario que exista un hilo que escuche y gestione los datos, bien ilustrándolos en tiempo real por consola o interfaz, o bien almacenándolos en un archivo en el disco duro para ser estudiado al finalizar la aplicación.

Esta primera idea supone una carga adicional a los recursos del NXT al añadir un hilo que se encarga específicamente de esta tarea y puede intervenir directamente en el aumento de los retardos de las comunicaciones o del propio rendimiento del robot. Es por esto que se decide analizar las dos alternativas que propone y proporciona las librerías de LeJOS.

- DataLogger: DataLogger es una clase implementada con la finalidad de almacenar valores de tipo float en un vector y transmitirlos a la aplicación DataViewer ofrecida por LeJOS en el software para PC.

Figura 67. Dataviewer.

De esta manera no se consumen tantos recursos en la ejecución como con la creación de un hilo específico. Pero la primera objeción del uso de DataLogger es la limitación del tipo de datos que se pueden almacenar (tan sólo tipo float) y también del tamaño del propio vector que aunque se puede configurar mediante el constructor, está limitado por la memoria disponible en el NXT. El medio de transmisión de los datos lo elije el usuario (Bluetooth o USB) al iniciar la aplicación desde el PC.

- La otra opción y según el estudio la más factible, es crear un archivo sencillo de texto en el NXT y escribir en él los datos. De esta manera no se está limitando el log a

ningún tipo de datos como en el caso anterior. Este archivo aparecerá entre los contenidos en el robot como el propio programa y el acceso al mismo se puede realizar también desde la aplicación Dataviewer del PC mediante Bluetooth o USB.

Sería interesante una función de transferencia de archivos completos entre el NXT y el PC y de este modo incorporarlo al propio programa y poder prescindir de aplicaciones externas para esto, pero hasta la fecha LeJOS no incorpora esta opción.

3.4. Concurrencia en Java. Hilos de ejecución.

Antes de comenzar con el estudio del movimiento de los NXT se requiere un estudio en la metodología de la programación de las aplicaciones a desarrollar.

Para la mayor parte de las aplicaciones programadas ha sido necesario el conocimiento y el manejo de varios hilos de ejecución. Por ejemplo, si la idea es que el robot siga un comportamiento o una rutina y a causa de un cambio externo al robot y detectado por el ordenador, se desea modificar el comportamiento del mismo, se necesitará al menos dos hilos de ejecución en el robot. Uno que lleve a cabo la rutina y otro que cada pequeño intervalo de tiempo escuche al ordenador para saber si debe cambiar su rutina debido a algún cambio externo a él. Algo parecido se desarrolló en la aplicación que se describe a continuación.

3.5. Simulador de corrección de trayectorias en tiempo real

La idea de esta aplicación es la simplificación del planteamiento siguiente:

Se supone un determinado escenario con objetos fijos y móviles, controlado y discretizado mediante una cámara cenital conectada a un ordenador. En un extremo del escenario se sitúa uno o varios robots NXT que deberán cruzar al otro lado evitando cualquier obstáculo que detecten y ayudados por la información capturada por la cámara.

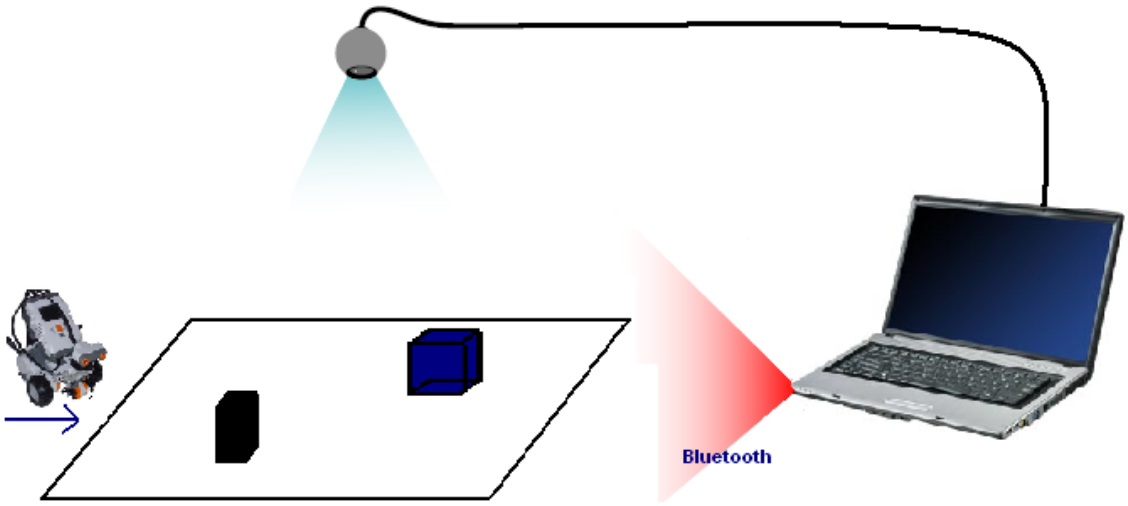


Figura 68. Escenario simulado.

La simulación se centra en las acciones y el manejo de los hilos ante la comunicación por Bluetooth entre el ordenador y los robots. El escenario se discretiza y cada robot se mueve de coordenada en coordenada. Ante la detección desde la cámara cenital de riesgo de colisión del NXT con un obstáculo, el ordenador debe comunicarle hacia qué nueva coordenada del escenario debe dirigirse para evitar la colisión.

Planteado el problema se estudió los hilos necesarios para la simulación y se siguió el siguiente esquema:

Figura 69. Esquema de hilos de ejecución.

En el PC, se crean, a raíz del hilo principal, un hilo de Conexión, un hilo de Escucha por Teclado y un hilo de Escucha Robots.

- Hilo de Conexión: es el encargado de realizar la conexión del PC con todos los robots y de finalizarla cuando acabe el programa.

- Hilo de Escucha por Teclado: la simulación de los eventos que puede capturar la cámara se realiza por entrada de teclado mediante comandos sencillos dónde mediante un número se especifica el robot NXT al que se hace referencia y una letra define la nueva coordenada destino hacia donde debe dirigirse. Este hilo se encarga de leer la entrada del teclado y enviar la información al robot correspondiente.

- Hilo de Escucha Robots: Este hilo rastrea a una alta frecuencia los mensajes que los NXT pueden haber enviado al PC y los muestra por consola. Dado que la simulación no es gráfica, estos mensajes son esenciales para comprobar la buena gestión de tareas.

Por otro lado, en cada NXT se crea también un hilo encargado únicamente de la Conexión, otro que Escuche al PC y otro que simule el movimiento.

- Hilo de Conexión (NXT): gestiona hasta el final la conexión vía Bluetooth con el PC.

- Hilo Escucha PC (NXT): este hilo es el que está atento a las posibles órdenes que puede recibir el robot desde el PC. Y realiza las funciones consecuentes que de detallarán más adelante.

- Hilo Simulador de traslación (NXT): simula el tiempo que conlleva que el robot se dirija de una coordenada a otra.

Si se analizan las diferentes situaciones que pueden darse en la simulación, una ejecución lineal sería mandar a un NXT a una coordenada dónde no se encuentre en ese momento, que el NXT se mueva hasta esa coordenada, llegue y se pare esperando la próxima orden. Pero la situación más interesante es que un robot se esté dirigiendo hacia una coordenada y antes de que llegue, desde el PC se le envíe otra coordenada distinta y deba cambiar el rumbo de inmediato. También es interesante el hecho de que reconozca si ya se encuentra en esa coordenada y por lo tanto no debe moverse o sí. Estas gestiones se incluyen en la función que ejecuta el hilo del NXT que escucha del PC. En caso de que se esté ejecutando el simulador de traslación y deba cambiar el rumbo, este hilo muere y se crea uno nuevo para dirigir al robot a la nueva coordenada.

Se ha comprobado el funcionamiento del simulador con 3 robots funcionando y conectados a la vez y los tiempos de transmisión fueron de 100ms de media.

3.6. Estudio del movimiento de los NXT

Los anteriores apartados están fuertemente ligados a la idea de que cualquier acción pueda ocurrir mientras los robots están en movimiento. Por tanto es muy importante conseguir un movimiento preciso, a la vez que controlado y manejable en tiempo real, puesto que debe estar sujeto a que se realicen cambios y responda rápidamente.

Leyendo los tutoriales que ofrece LeJOS se encuentran una serie de funciones de alto nivel que gestionan el movimiento de los NXT.

3.6.1. Tratamiento de las funciones de alto nivel

Estas son las funciones de alto nivel que incorporan las librerías de LeJOS :

- void forward() : para moverse hacia delante.
- Void backward() : para ir hacia atrás.
- Void stop() : para parar el motor.
- Void travel(float distance) : permite que el robot recorra cierta distancia especificada.
- Void steer(int turnRate, int angle) : el robot gira un ángulo especificado con un radio determinado por el valor de turnrate.

También incorpora funciones de navegación que permiten configurar y conocer la posición del robot en coordenadas x, y en cada momento. Se desarrollaron algunos programas de prueba haciendo que el robot recorriera una trayectoria circular y un rectángulo y las funciones respondieron con precisión sin embargo estas funciones no alcanzan algunas necesidades como por ejemplo que dado el robot en una posición gire en círculos respecto a un centro cualquiera. El hecho de que no ofrezcan libertad absoluta de movimiento obliga a gestionar el movimiento mediante funciones a bajo nivel.

3.6.2. Gestión de la trayectoria a bajo nivel

Cuando se habla de gestionar el movimiento del NXT a bajo nivel, se habla de controlar la velocidad de los motores mediante el voltaje que se les aplica, se habla de controlar recorridos y distancias mediante el conteo de los encoders.

LeJOS ofrece funciones para el tratamiento de los encoders y del voltaje de los motores:

- ResetTachoCount(): Esta función sustituye a la función nMotorEncoder[motor]=0, usada en RobotC para poner a cero el conteo de los encoders.

- GetTachoCount(): Se utiliza para conocer el valor de los encoders de un motor.

- Setpower(): Es la función que aplica el voltaje deseado al motor que se le especifique. Acepta un margen de 0 a 100.

Es importante acordarse de desactivar la opción de regular la velocidad de los motores que por defecto viene activada. Para esto tan sólo hay que llamar a la función regulateSpeed(false) para cada motor que sea necesario.

3.7. Estrategias del control cinemático.

Para el control de trayectorias se ha estudiado dos métodos geométricos diferentes: el control o seguimiento de trayectorias mediante la estrategia del punto descentralizado y el algoritmo de persecución pura.

3.7.1. Punto descentralizado

El control de posición por punto descentralizado se establece a partir de la posición y velocidad de un punto que está separado una distancia ***e*** del eje de tracción del robot.

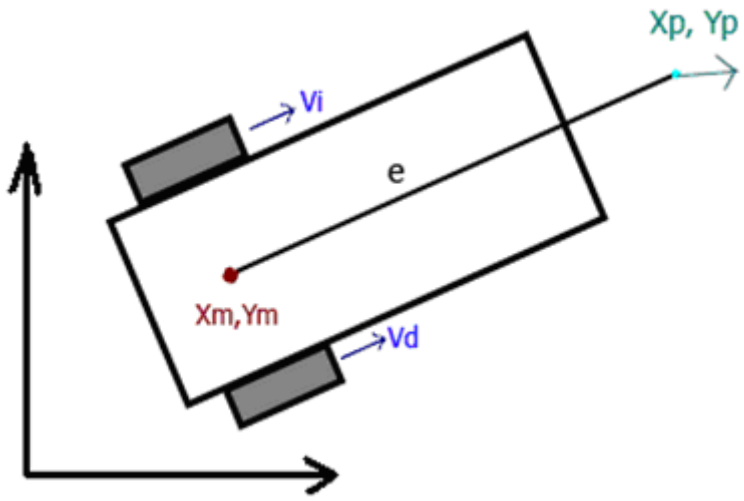


Figura 70. Punto descentralizado.

De este modo las coordenadas del punto descentralizado vienen definidas por:

$$\begin{bmatrix} \dot{x}_p \\ \dot{y}_p \end{bmatrix} = \begin{bmatrix} \dot{x}_m & + e\cos(\theta) \\ \dot{y}_m & + e\sin(\theta) \end{bmatrix}$$

Se calcula la derivada (velocidad) de ese punto:

$$\begin{bmatrix} \dot{x}_p \\ \dot{y}_p \end{bmatrix} = \begin{bmatrix} \dot{x}_m & - e\cos(\theta)\dot{\theta} \\ \dot{y}_m & + e\sin(\theta)\dot{\theta} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -e\sin(\theta) \\ 0 & 1 & e\cos(\theta) \end{bmatrix} \begin{bmatrix} \dot{x}_m \\ \dot{y}_m \\ \dot{\theta} \end{bmatrix}$$

Para conocer la posición del robot en todo momento se parte de la ecuación cinemática directa diferencial:

$$\begin{bmatrix} \dot{x}_m \\ \dot{y}_m \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos(\theta) & 0 \\ \sin(\theta) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ w \end{bmatrix} = \begin{bmatrix} \cos(\theta) & 0 \\ \sin(\theta) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1/2 & 1/2 \\ -1/2b & 1/2b \end{bmatrix} \begin{bmatrix} v_i \\ v_d \end{bmatrix}$$

Donde sustituyendo queda:

$$\begin{bmatrix} \dot{x}_p \\ \dot{y}_p \end{bmatrix} = \begin{bmatrix} 1 & 0 & -e\sin(\theta) \\ 0 & 1 & e\cos(\theta) \end{bmatrix} \begin{bmatrix} \dot{x}_m \\ \dot{y}_m \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -e\sin(\theta) \\ 0 & 1 & e\cos(\theta) \end{bmatrix} \begin{bmatrix} \cos(\theta) & 0 \\ \sin(\theta) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1/2 & 1/2 \\ -1/2b & 1/2b \end{bmatrix} \begin{bmatrix} v_i \\ v_d \end{bmatrix}$$

$$\begin{bmatrix} \dot{x}_p \\ \dot{y}_p \end{bmatrix} = \frac{1}{2b} \begin{bmatrix} b\cos(\theta) + e\sin(\theta) & b\cos(\theta) - e\sin(\theta) \\ b\sin(\theta) - e\cos(\theta) & b\sin(\theta) + e\cos(\theta) \end{bmatrix} \begin{bmatrix} v_i \\ v_d \end{bmatrix}$$

Las velocidades de las ruedas se pueden obtener despejando de la ecuación anterior:

$$\begin{bmatrix} v_i \\ v_d \end{bmatrix} = \frac{1}{e} \begin{bmatrix} e\cos(\theta) + b\sin(\theta) & e\sin(\theta) - b\cos(\theta) \\ e\cos(\theta) - b\sin(\theta) & e\sin(\theta) + b\cos(\theta) \end{bmatrix} \begin{bmatrix} \dot{x}_p \\ \dot{y}_p \end{bmatrix}$$

Para el desarrollo del control cinemático de los NXT se ha utilizado la siguiente ecuación:

$$\begin{bmatrix} \dot{x}_p \\ \dot{y}_p \end{bmatrix} = \begin{bmatrix} k_{v_x}\dot{x}_{ref} \\ k_{v_y}\dot{y}_{ref} \end{bmatrix} + \begin{bmatrix} k_{p_x} & 0 \\ 0 & k_{p_y} \end{bmatrix} \begin{bmatrix} x_{ref} - (x_m + e\cos\theta) \\ y_m - (y_m + e\sin\theta) \end{bmatrix}$$

Entre las ventajas de la estrategia del punto descentralizado cabe destacar que no es necesario conocer la trayectoria completa para aplicarla sino que al mismo tiempo que se calcula la velocidad necesaria para que el robot alcance la siguiente posición, se calcula el siguiente punto en función de la posición actual donde se encuentra el robot en ese momento. Lo cual ofrece precisión y una continuidad uniforme en el movimiento.

3.7.2. Persecución Pura.

A diferencia de la estrategia del punto descentralizado, para aplicar el algoritmo de persecución pura es necesario conocer la trayectoria previamente.

Dada una posición (X_R,Y_R) y un punto objetivo (Xob,Yob) se puede analizar el recorrido como sigue:

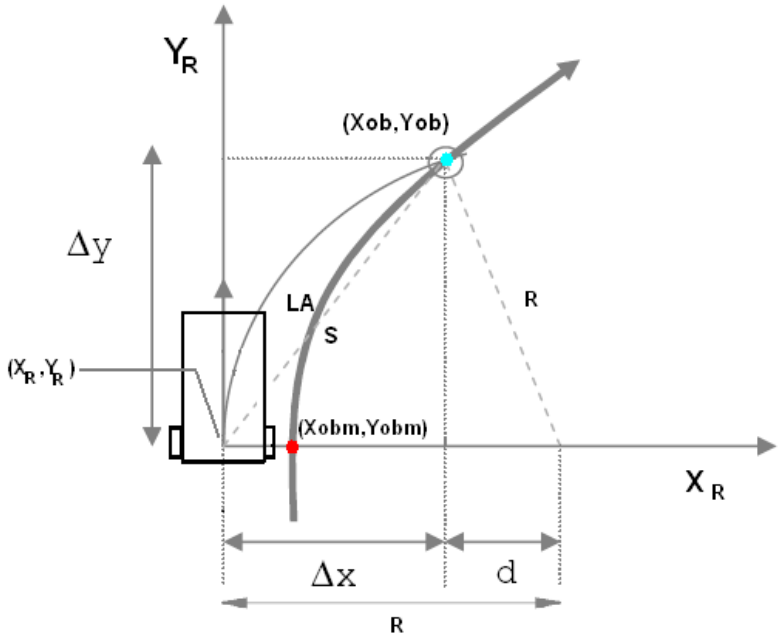


Figura 71. Persecución pura.

Para una velocidad V constante, en cada período de control se calcula el punto (X_{obm},Y_{obm}) del camino que esté más próximo al robot (X_R,Y_R) y se elige el punto objetivo (X_{ob},Y_{ob}) situado a una distancia fija S.

Partiendo de la ley de los catetos donde:

$$\Delta x^2 + \Delta y^2 = LA^2$$

Del esquema se puede deducir:

$$\Delta x + d = R \rightarrow d = R - \Delta x$$

Y sustituyendo y desarrollando se puede calcular la curvatura (γ) como:

$$(R - \Delta x)^2 + \Delta y^2 = R^2$$

$$R^2 - 2R\Delta x + \Delta x^2 + \Delta y^2 = R^2$$

$$2R\Delta x = LA^2 \rightarrow R = \frac{LA^2}{2\Delta x}$$

$$\gamma = \frac{1}{R} = -\frac{2\Delta x}{LA^2}$$

La distancia LA estará definida en función de las coordenadas actuales y del punto objetivo:

$$LA = \sqrt{(x_{ob} - x_R)^2 + (y_{ob} - y_R)^2}$$

$$\Delta x = (x_{ob} - x_R)\cos\theta + (y_{ob} - y_R)\sin\theta$$

Y, por último, se calculan las velocidades de cada rueda como sigue:

$$v_d = V + \frac{b}{2}\omega = V + \frac{b}{2}\frac{V}{r} = V\left(1 + \frac{b}{2}\frac{1}{r}\right)$$

$$v_d = V\left(1 + \frac{b}{2}\gamma\right)$$

$$v_i = V - \frac{b}{2}\omega = V - \frac{b}{2}\frac{V}{R} = V\left(1 - \frac{b}{2}\frac{1}{R}\right)$$

$$v_i = V\left(1 - \frac{b}{2}\gamma\right)$$

De esta manera se aplican las respectivas velocidades a los motores para que el robot siga la trayectoria que se propone.

El hecho de deber conocer y si no, en peor caso, tener que calcular los puntos del camino que debe recorrer previamente el robot a la ejecución de las acciones de control cinemático, aumenta el coste y con ello el tiempo de la ejecución. Sin embargo ofrece la ventaja de poder asegurar que el robot pase por puntos clave y que siga el camino de una forma más estricta.

3.8. The ring: tratamiento de una trayectoria circular, constante, variable en tiempo real y compartida por múltiples robots en un mismo escenario.

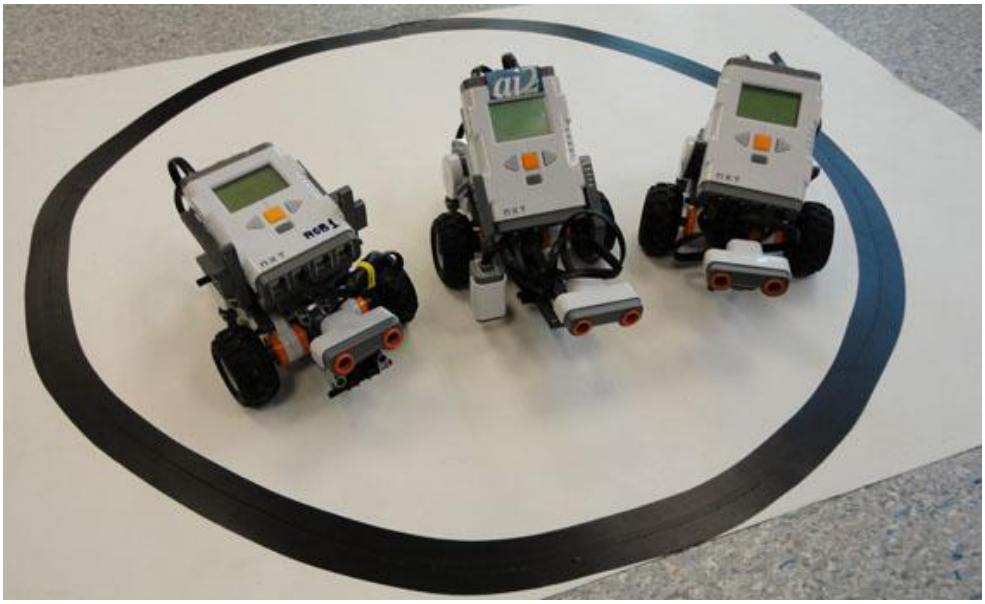


Figura 72. Presentación The ring

3.8.1. Descripción de la aplicación y características

La aplicación The ring consiste en la gestión del movimiento y de la trayectoria circular que comparten tres robots NXT.

Los tres robots siguen una misma trayectoria circular, separados por una distancia uniforme de seguridad.

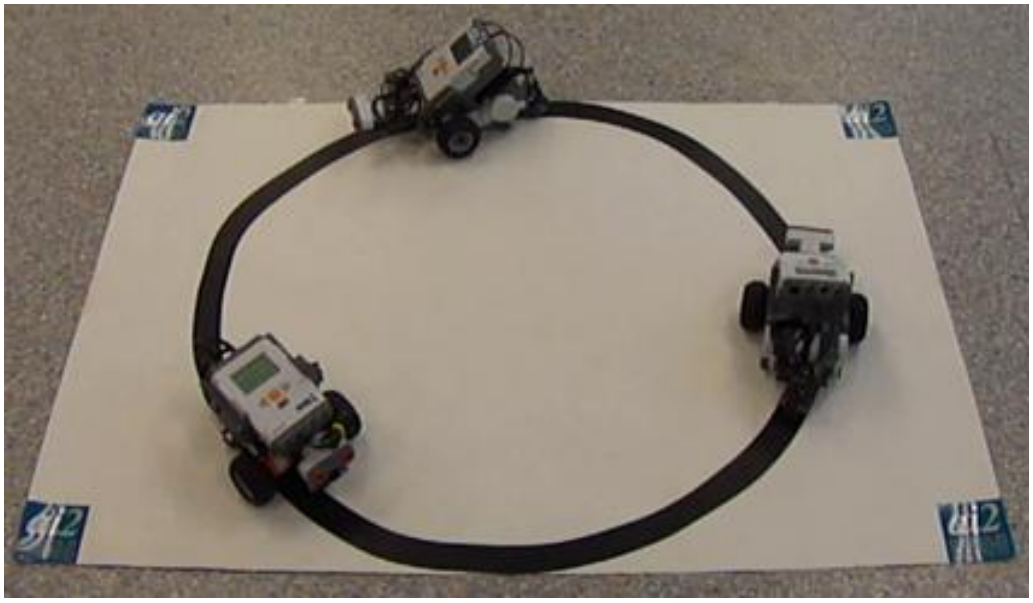


Figura 73. The ring

Existe una comunicación vía Bluetooth entre los NXT y el PC en todo momento. Desde el PC se monitorizan los movimientos y en tiempo real es posible cambiar el radio de la trayectoria circular que siguen los robots y también el centro de la misma. Esto provoca un desplazamiento repentino de los robos hacia su nueva posición y un gran riesgo de colisión entre ellos que se gestiona gracias al sensor de sónar y a una estrategia de evitación de obstáculos.

3.8.2. Metodología y desarrollo de la aplicación

La aplicación requiere al menos tres hilos de ejecución en cada robot. Uno para la conexión, otro que esté atento a las órdenes que puede recibir del PC y otro que lleva a cabo el movimiento circular.

- El hilo de conexión inicia la comunicación vía Bluetooth con el PC y la mantiene hasta el final de la aplicación.

- El hilo de movimiento contiene la estrategia de gestión de trayectorias y es el encargado del movimiento circular del NXT.

- El hilo lector del PC escucha cada cierto período de tiempo si el PC ha enviado algún dato al robot. Un cambio de radio o de centro de trayectoria. Y en ese caso actúa sobre el hilo de movimiento para que corrija su trayectoria.

La relación entre los threads en ejecución es la que sigue:

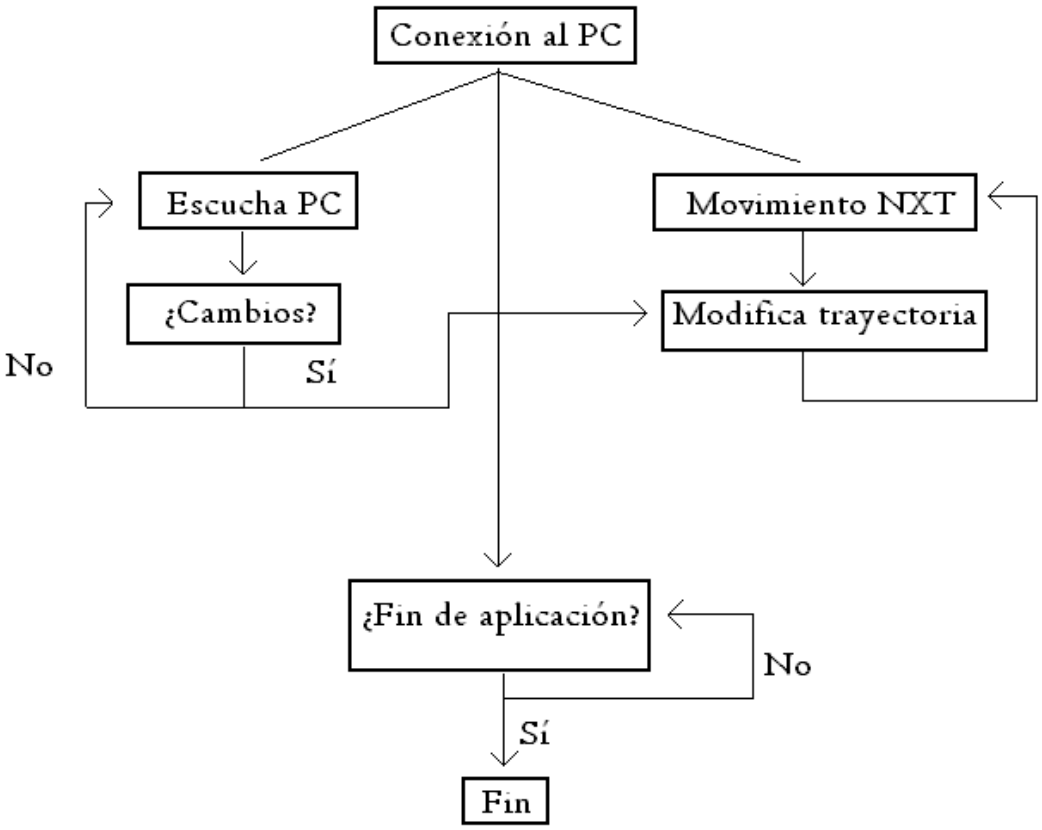


Figura 74. Relación entre los threads.

Por otro lado, en el PC también se ejecutan tres hilos pero con distintas funciones.

- El hilo de conexión tiene la función de sincronizar y conectar con los tres NXT.

- El hilo de cambios espera a que se produzca el cambio de radio o centro de la trayectoria circular a través del usuario mediante la interfaz de la aplicación. Y en ese caso envía la información a los NXT.

- El hilo monitor recibe las coordenadas exactas de los NXT y si se selecciona la opción de abrir simulador, muestra en una imagen los movimientos en tiempo real que van siguiendo los robots.

De ese modo se gestionan los eventos que pueden ocurrir en la aplicación.

Para la gestión de trayectoria se ha probado con las estrategias del punto descentralizado y de persecución pura, y para la aplicación definitiva se ha optado por utilizar únicamente la del punto descentralizado ya que en las pruebas ofrecía mayor corrección frente a posibles errores.

En relación con la detección y evitación de obstáculos, el sensor de sónar instalado en los tres NXT, rastrea en todo momento si existe algún obstáculo en frente y en caso de que esté lo suficientemente cerca como para que haya un riesgo de colisión, a la gestión de la trayectoria se le añade un módulo de control cinemático que modifica en ese caso dicha trayectoria.

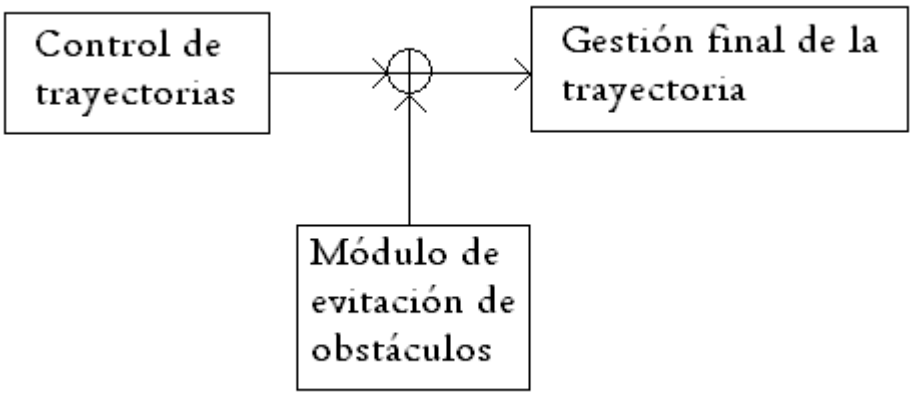


Figura 75. Gestión del módulo de obstáculos

Este módulo de control de obstáculos se acentúa en el resultado de la trayectoria cuanto más cerca se detecta el objeto, hasta el punto de que si casi va a colisionar, hace que el robot se detenga. Cuando no se detecta ningún obstáculo, el módulo de obstáculos es nulo y la gestión de la trayectoria del punto descentralizado pasa directamente a la gestión final.

Lo que ocurre con este esquema es que la gestión individual del seguimiento de trayectoria por punto descentralizado y del módulo de evitación de obstáculos, hace que, durante el tiempo en el que un robot se encuentra evitando un obstáculo, el control de trayectoria continúa por su parte generando la trayectoria y cuando la evitación de obstáculos desaparece y vuelve a tener el control total sobre la gestión final, el punto donde se debería encontrar el robot puede estar lejos de su posición actual. Lo cual hace que el robot tome el camino más corto hasta volver a reengancharse a la trayectoria.

3.8.3. Sistema de archivos, interfaz y ejecución.

El sistema de archivos de la aplicación consta de cuatro proyectos. Tres de ellos se cargan en los NXT y el cuarto se ejecuta en el ordenador.

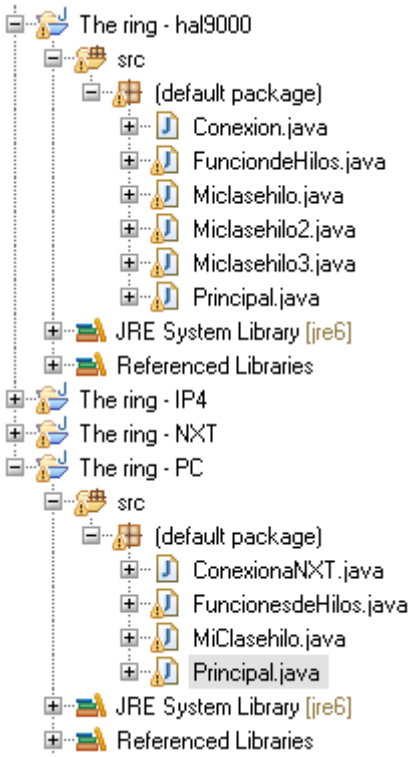


Figura 76. Proyectos y clases.

Cada proyecto de los robots contiene seis clases .java que se comunican entre sí. Los nombres de los proyectos hacen referencia al nombre del robot al que se debe cargar ya que cada uno incluye el nombre en la clase Conexion.java que es necesario para realizar la conexión Bluetooth.

Tanto para mandar los proyectos a los NXT como para ejecutar el del PC, hay que hacerlo sobre la clase Principal.java ya que es la que incluye la función main.

Una vez cargados los programas en los robots, se ejecuta la aplicación del PC y aparece lo siguiente:

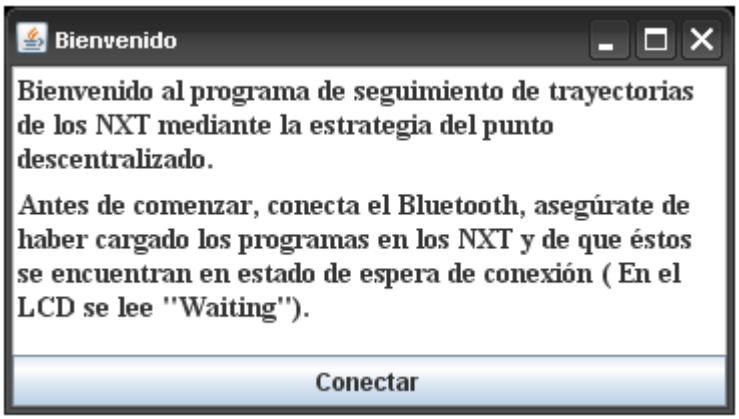


Figura 77. Bienvenida

Figura 78. LCD NXT waiting

Si está todo listo se pulsa Conectar, y la aplicación irá conectando vía Bluetooth con cada uno de los NXT. Cuando todos estén listos el siguiente paso será calibrarlos:

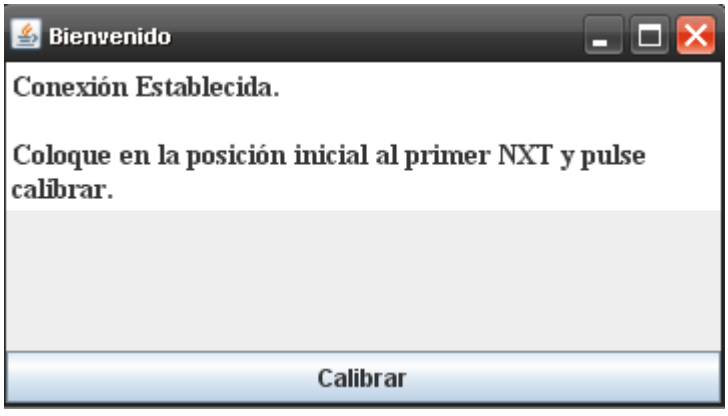


Figura 79. Interfaz, paso 1.

Por defecto comienza el NXT hal9000 en la posición 335,0 (en x,y mm) de la trayectoria circular y con una orientación de 180°. Se pulsa calibrar y el robot comenzará a seguir la trayectoria hasta haber recorrido un ángulo de 240° y se parará. En el PC aparecerá la siguiente pantalla:

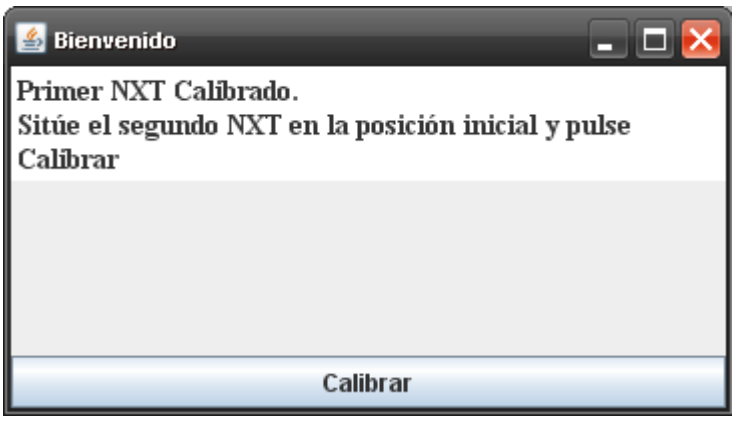


Figura 80. Interfaz, paso 2.

Se coloca al robot con el nombre IP4 en la misma posición y orientación inicial que hal9000 y se pulsa "Calibrar", con lo que IP4 seguirá la misma trayectoria circular y se detendrá cuando haya recorrido 120° de la circunferencia, avisando al PC de que ya está calibrado.

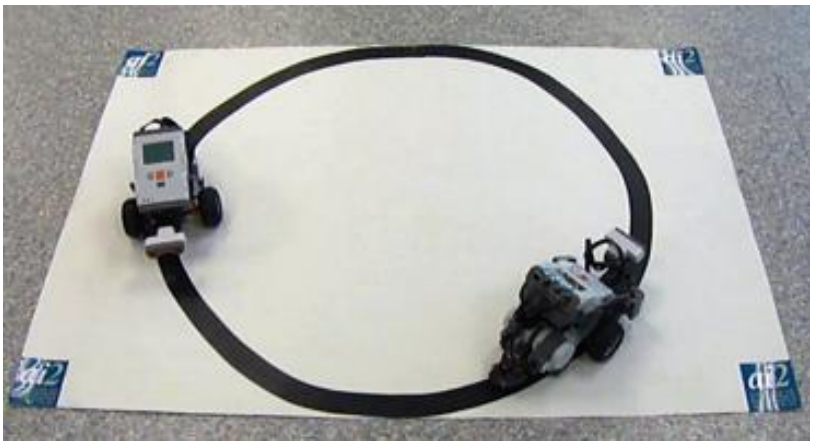


Figura 81. hal9000 e IP4 calibrados.

Por supuesto hay que verificar visualmente que los dos robots circulan bajo la misma trayectoria y llegan a su posición en la orientación adecuada, en caso contrario no estarían bien calibrados y habría que comenzar de nuevo.

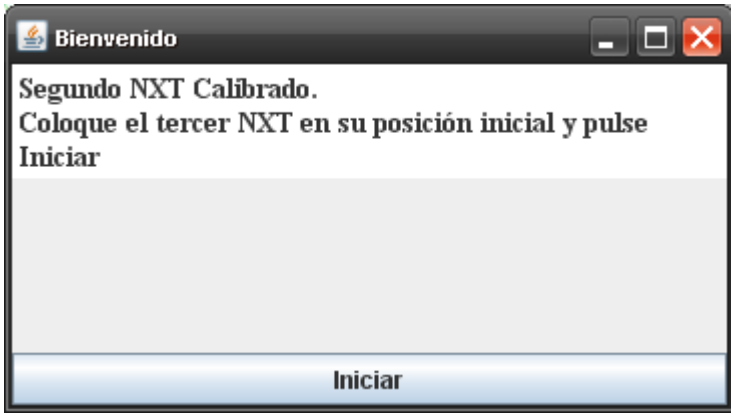


Figura 82. Interfaz Iniciar

Se coloca al llamado NXT en la misma posición inicial que el resto y se pulsa "Iniciar". Los tres robots comienzan a girar a la misma velocidad y siguiendo la trayectoria circular. En el PC aparece la siguiente interfaz:

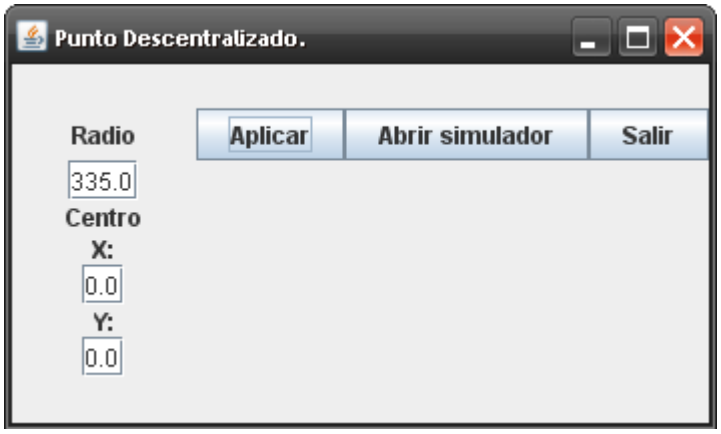


Figura 83. Interfaz

Para cambiar de radio y/o de centro la trayectoria circular, hay que editar los campos correspondientes y al pulsar "Aplicar" se les envía la información a los NXT por Bluetooth.

Si se pulsa en "Abrir simulador" aparece una ventana donde se van dibujando las posiciones de los robots junto a la trayectoria circular que deberían estar siguiendo en ese momento. Para un robot, un cambio en la dimensión del radio de la trayectoria quedaría registrado así:

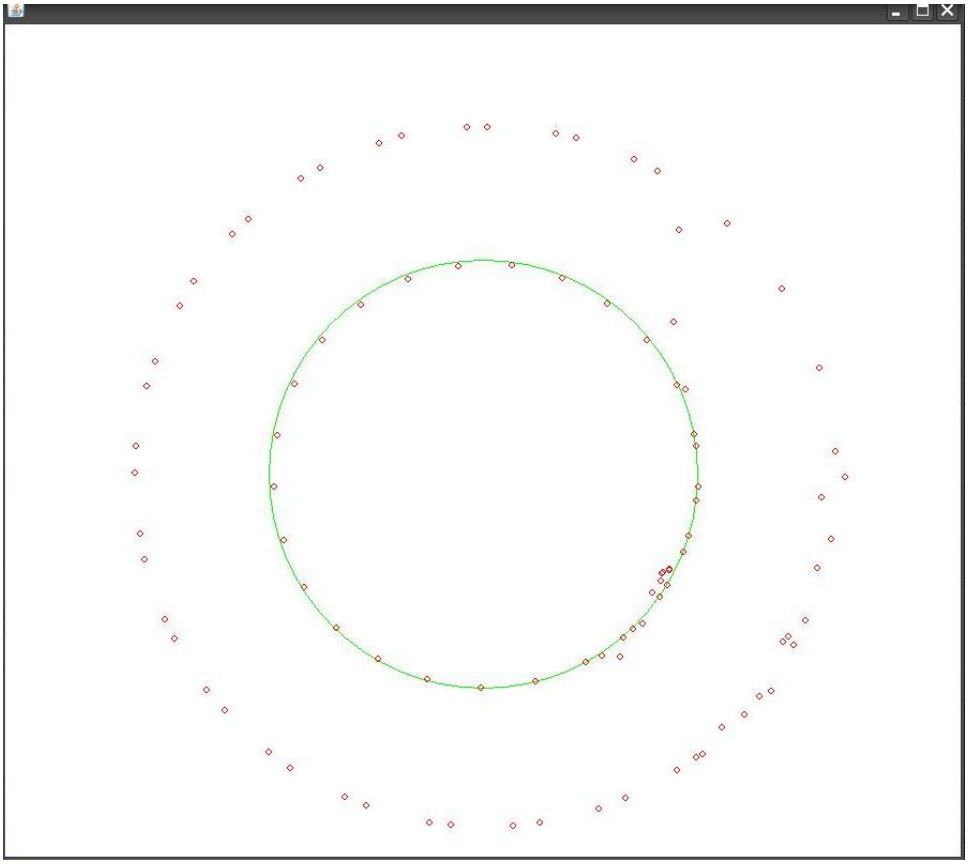


Figura 84. Simulador

En este caso se ha reducido el radio y mantenido el centro de la circunferencia. Inicialmente el robot recorría la circunferencia exterior y se redujo el radio a la circunferencia verde. Puede observarse claramente la transición del radio exterior al interior que tubo que realizar el robot.

Para distinguir en el simulador a los tres NXT, cada uno se representa con un color. Un cambio de centro con los tres robots queda registrado de la siguiente manera.

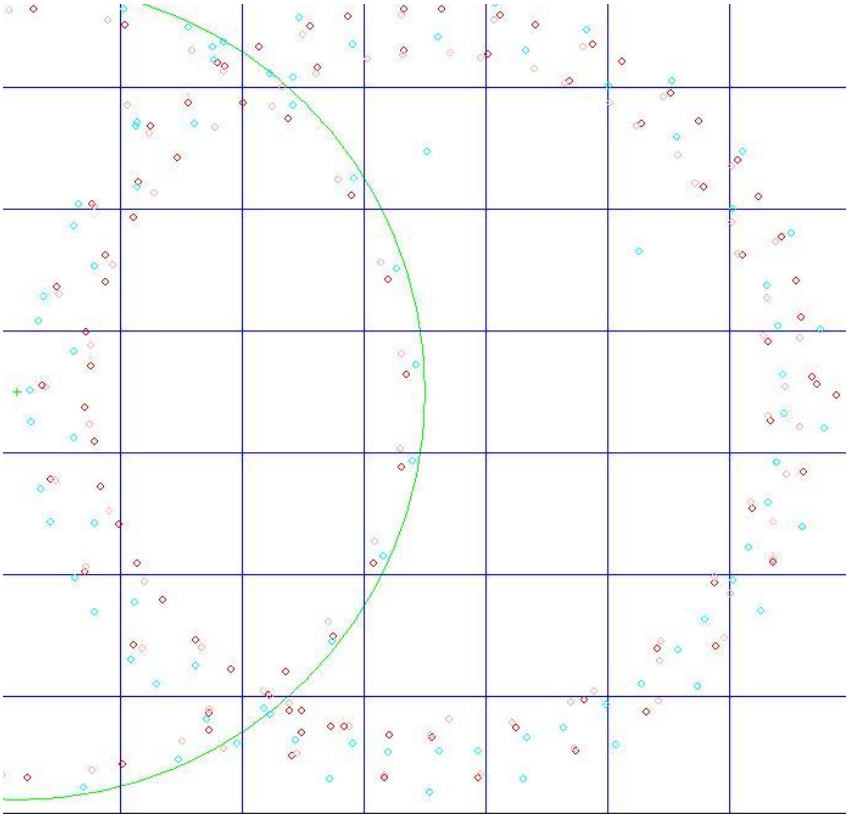


Figura 85. Simulador centro

Para terminar la aplicación en cualquier momento tan sólo hay que pulsar el botón “Salir” de la interfaz.

Para una demostración de ejecución de la aplicación, se adjunta un vídeo demostrativo con el nombre The ring.

3.8.4. Problemas encontrados y soluciones

El mayor problema en el desarrollo de la aplicación ha sido la precisión. Los motores admiten un rango de voltaje de 0 a 100 pero según la superficie sobre la que se sitúe a los NXT se debe acotar más la velocidad máxima (si la superficie es pulida) o menos (si no lo es) porque si no el mínimo patinaje en las ruedas hace perder precisión y que el resultado no sea el pretendido.

Por supuesto la distancia entre las ruedas hay que comprobarla de vez en cuando porque pueden haber ligeros desajustes que distorsionen el recorrido y sobra decir que la batería debe estar cargada para que no influya en la sincronía de los robots.

Cabe mencionar también que al principio cada robot comenzaba desde una posición inicial de la trayectoria de la circunferencia sin calibración, pero hubo que sustituir este método por el de calibración por inexactitud.

3.9. Gestión de la visión

Bajo la idea de plantear una aplicación parecida a la del problema descrito en la sección 3.5, fue necesario estudiar la manera de capturar, mediante una cámara, un escenario determinado y saber reconocer ciertas formas y colores. A los NXT se les colocó un señuelo en forma de triángulo isósceles de color negro para facilitar su reconocimiento desde la cámara cenital. El color del triángulo facilitó su distinción entre el escenario, y la forma, su orientación.

Figura 86. Robot con señuelo.

Para la gestión de la visión se optó por usar la librería OpenCV 1.0 para JAVA. Se puede descargar desde http://www.ubaa.net/shared/processing/opencv/ donde además se incluyen ejemplos y manuales. Para la instalación tan sólo es necesario descomprimir e incluir la librería OpenCV.jar en el Java Build Path del proyecto.

Se ha desarrollado una pequeña aplicación que captura imágenes con una alta frecuencia desde una webcam y las vuelca a un frame que se muestra y se refresca por pantalla. Luego, por cada captura, se recorre toda la imagen buscando píxels del color

del señuelo colocado a los NXT y se resaltan en la imagen volcada para indicar dónde están situados.

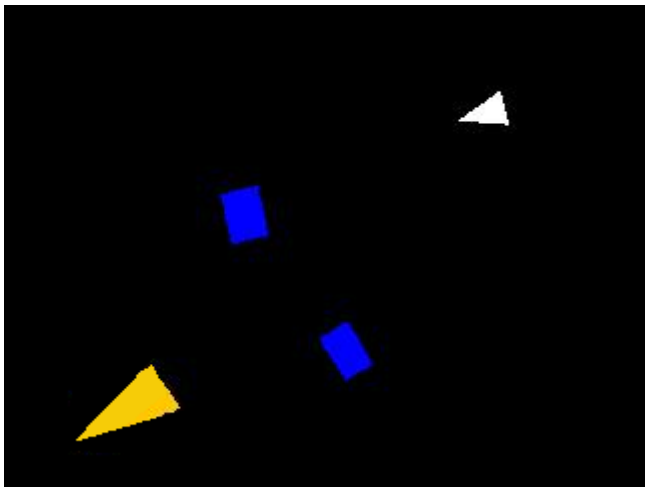


Figura 87. Reconocimiento de objetos

Para conocer la orientación de cada objeto se calcula el punto más alejado al centroide que forma parte del perímetro del objeto. Se dibuja una línea del centroide a ese punto, que es la que marca el ángulo de orientación.

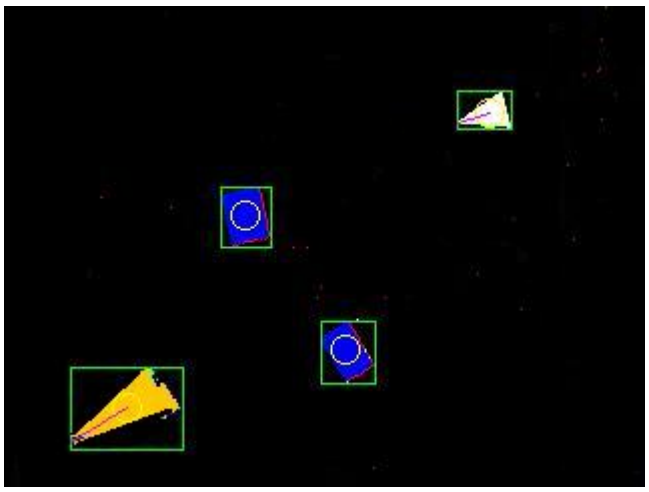


Figura 88. Centroides y orientación.

3.10. El laberinto: generación de una trayectoria a través del reconocimiento de un escenario mediante una cámara cenital.

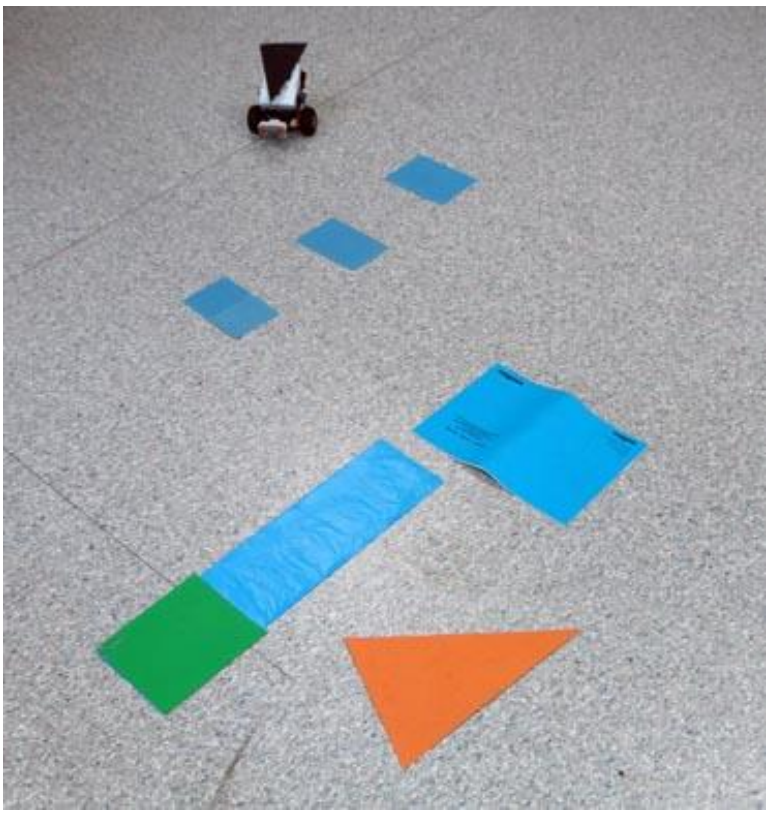


Figura 89. Laberinto

3.10.1. Descripción

La aplicación del laberinto consiste en, dado un escenario, el reconocimiento mediante una cámara cenital de la posición inicial de un NXT, de los obstáculos y de la posición final que debe alcanzar el robot, generar una trayectoria que debe recorrer el NXT para llegar a esa posición final evitando los obstáculos.

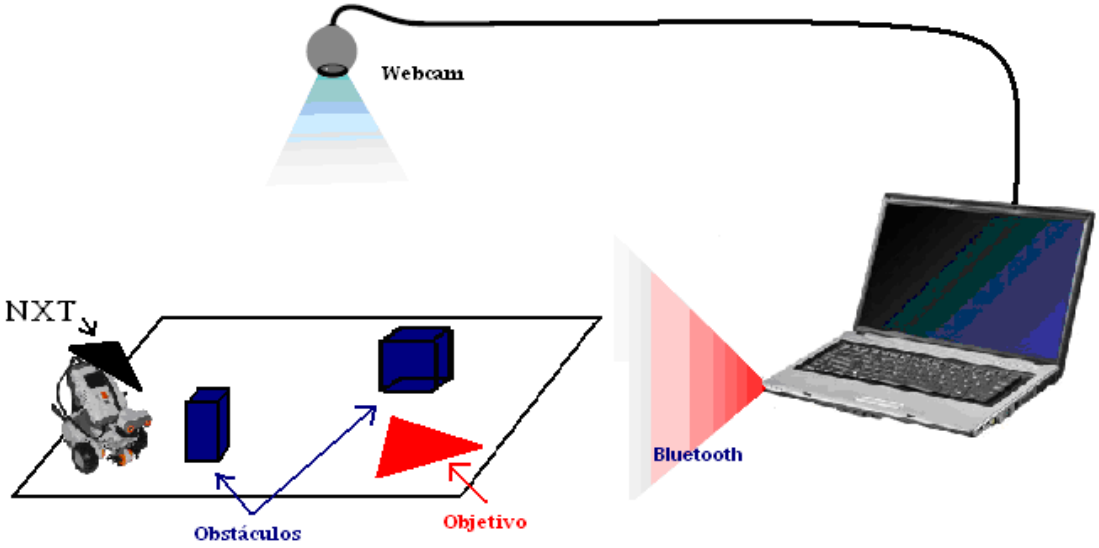


Figura 90. Esquema de la aplicación

La cámara cenital está conectada al PC que gestiona la aplicación y que al mismo tiempo se comunica mediante Bluetooth con el robot para enviarle la trayectoria que debe seguir.

3.10.2. Metodología y desarrollo de la aplicación.

En el caso de esta aplicación, la programación es bastante lineal. En el PC se ejecuta el hilo principal que conecta con el NXT y con la cámara, realiza las operaciones de reconocimiento de formas, crea la trayectoria y se la envía al robot. Tan sólo se crea un hilo secundario para crear, mostrar y refrescar el frame que contiene las imágenes que se van capturando por la cámara.

Del mismo modo, en el NXT se ejecutan tan sólo dos hilos, uno para gestionar la conexión Bluetooth con el PC y otro para realizar la trayectoria.

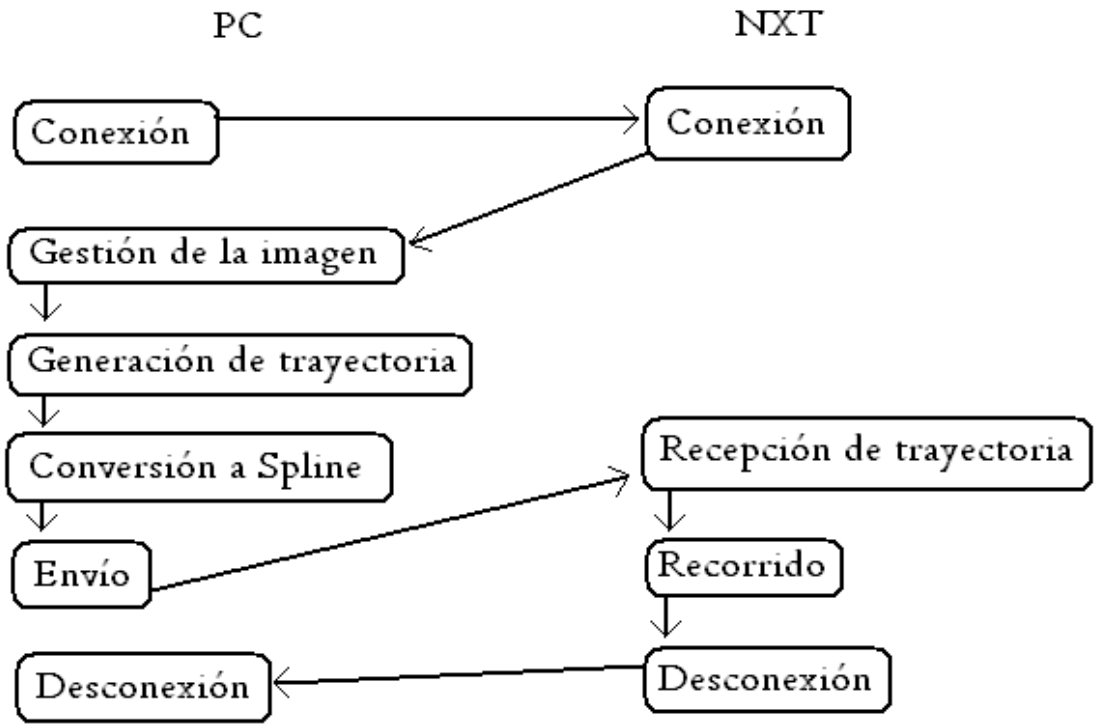


Figura 91. Esquema de ejecución.

Para el reconocimiento de formas mediante OpenCV, la imagen se discretiza aislando los señuelos del resto del escenario y diferenciándolos de si son obstáculos o son los señuelos de inicio o final de posición del robot. Para los señuelos de posición inicial y de posición final, se calculan los centroides y su orientación. Y para los obstáculos, se convierte la imagen a formato binario para enviarla al algoritmo de generación de trayectorias. Este algoritmo recibe la imagen binaria y la posición inicial y final del robot, y mediante el parámetro de raster, subdivide la imagen para conocer por qué coordenadas es posible pasar o no. Cada paso supone un coste que garantiza que la trayectoria que se calcula es la más eficiente.

Cuando se obtiene la trayectoria, se envía al módulo de transformación a Spline para que los movimientos del robot no sean bruscos y tenga una circulación más suave.

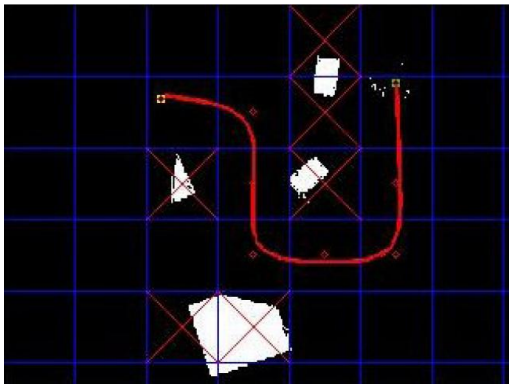


Figura 92. Algoritmo de generación de trayectoria + Conversión a Spline

Luego se transforman los píxels en mm mediante una matriz de transformación calculada específicamente para la cámara y el escenario que se ha estudiado, y se envía la trayectoria por Bluetooth al NXT.

Para la gestión de la trayectoria se utiliza la estrategia a bajo nivel del punto descentralizado, pero cuando el robot se sitúa en su posición final, la aplicación hace uso de las funciones de alto nivel proporcionadas por LeJOS para conseguir que se oriente según la marca del señuelo.

3.10.3. Sistema de archivos, interfaz y ejecución.

El sistema de archivos en este caso tan sólo consta de dos proyectos, uno para ejecutar en el PC y otro para el NXT que se vaya a utilizar:

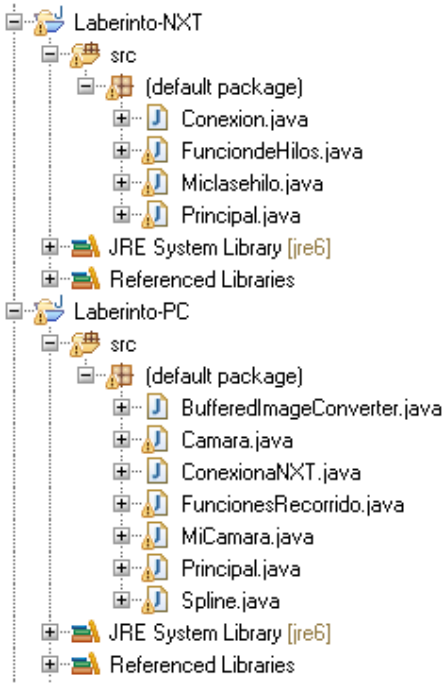


Figura 93. El laberinto: sistema de archivos

Hay que tener en cuenta que el nombre del robot es determinante para que la aplicación conecte con el PC y por eso es necesario comprobar que se corresponde con el que figura en las clases Conexion.java de Laberinto-NXT y en ConexionaNXT.java de Laberinto-PC. Las clases que contienen la función main en cada proyecto y que hay que ejecutar y cargar son Principal.java.

Al iniciar la aplicación en el PC, se muestra un mensaje de bienvenida que recuerda que es necesario haber cargado el programa al NXT y que éste se encuentre en estado de espera.

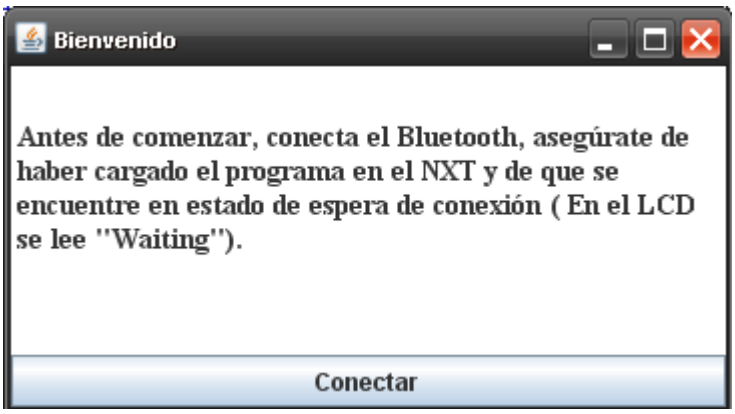


Figura 94.bienvenida

Hay que pulsar en conectar y una vez se establece la conexión, iniciar la cámara.

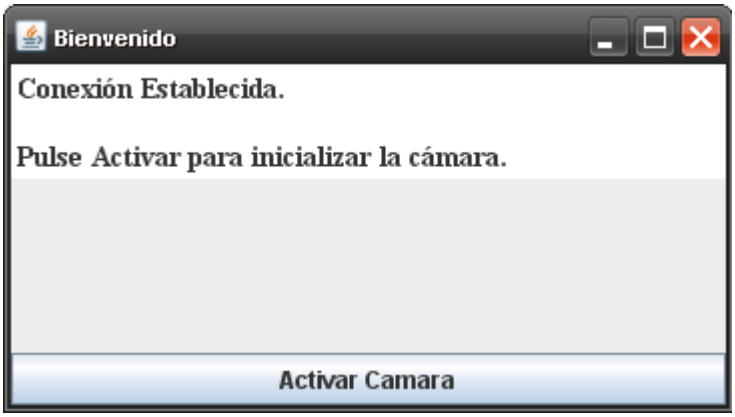


Figura 95. Conexión establecida

Se abre una nueva ventana donde se muestra un frame que muestra en tiempo real el resultado de discretizar y reconocer el escenario que captura la cámara. El señuelo del robot debe ser triangular y de color negro y el centroide del triángulo debe coincidir con el eje central de las dos ruedas motoras. Los señuelos de los obstáculos deben ser de color azul o verde claro y el señuelo que indica la posición final debe ser con forma triángulo para poder calcular la orientación final y de color naranja o rojo.

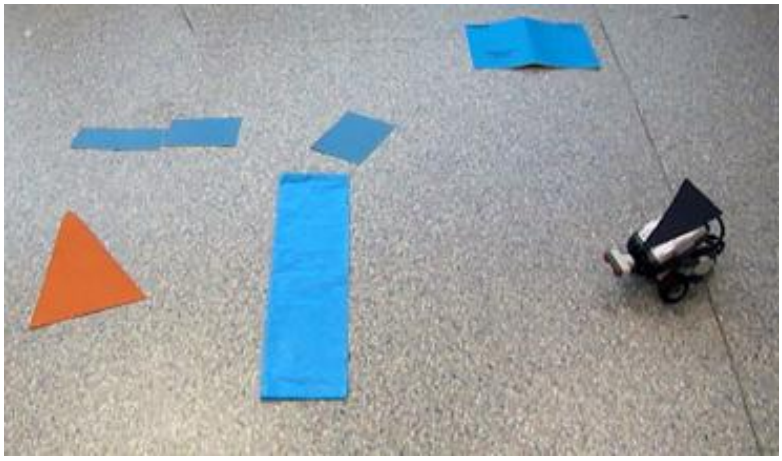


Figura 96. Ejemplo de escenario

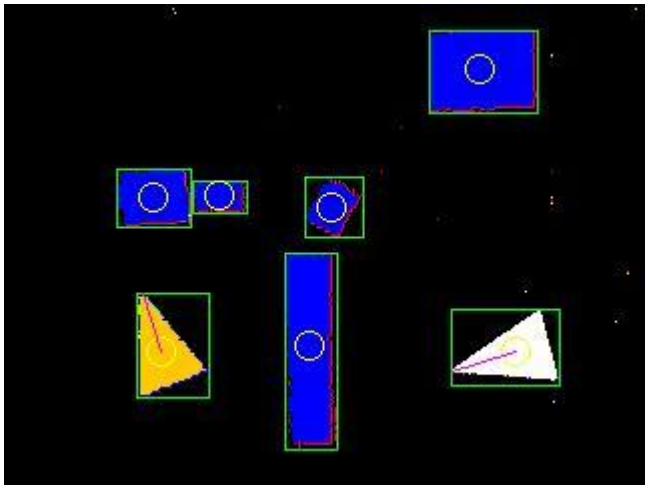


Figura 97. Escenario discretizado

A tiempo real se muestra por consola las coordenadas en mm. y la orientación de la posición inicial y final, por si en tiempo real se decide cambiar alguna de ellas.

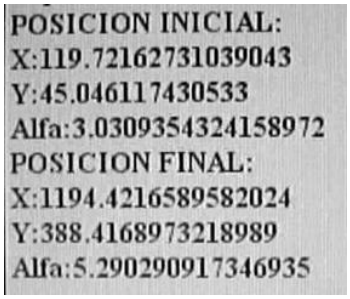


Figura 98. Consola

Al pulsar en capturar, se muestra el escenario capturado discretizado y en la ventana principal aparecen las posiciones definitivas de las posiciones de inicio y fin. Mientras tanto en el mismo directorio donde se encuentra el proyecto de la aplicación que se ejecuta en el PC, se almacena la captura final en modo de imagen

con nombre original.jpg, la imagen discretizada como discretizada.jpg y la imagen binaria que contiene sólo los obstáculos en blanco como binaria.jpg.

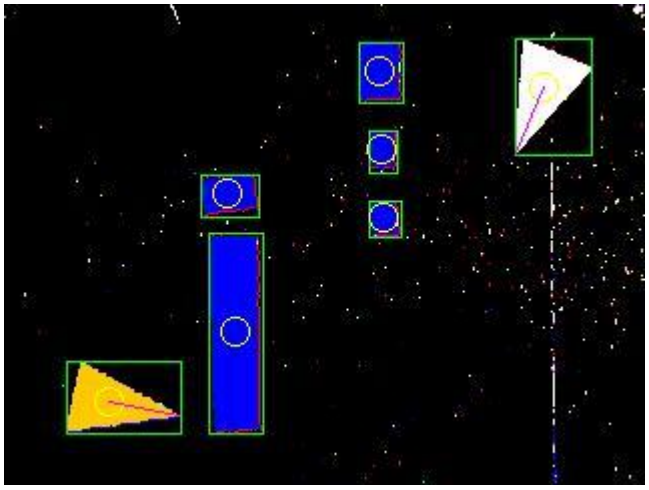


Figura 99. Original.jpg

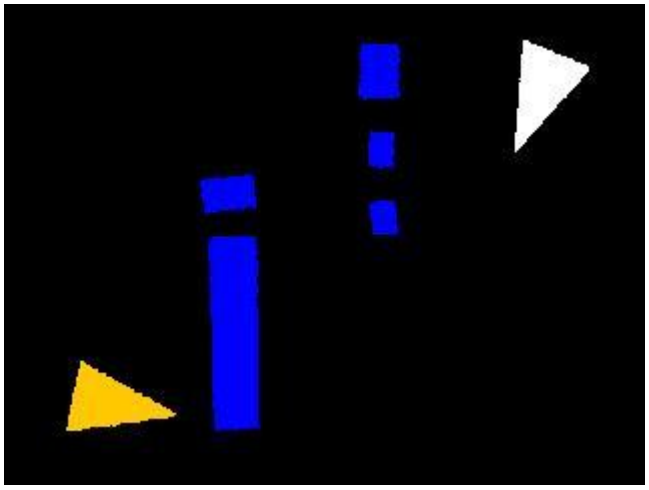


Figura 100. Discretizada.jpg

Figura 101. Binaria.jpg

El paso siguiente es pulsar calcular para enviar la imagen binaria al algoritmo de generación de trayectorias. Este proceso más la subdivisión en coordenadas más consecutivas y la transformación a Spline puede tardar unos segundos. Al terminar se muestran las coordenadas por consola y se almacena una imagen con la simulación del recorrido para comprobar que es correcto antes de enviarlo al robot.

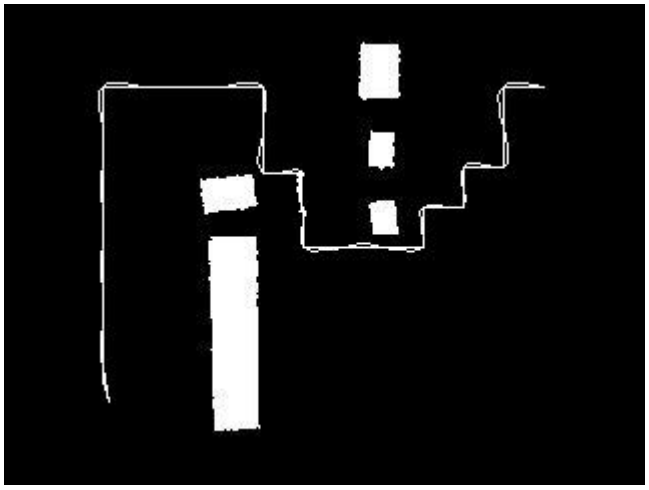


Figura 102. Recorrido.jpg

Si todo es correcto hay que pulsar enviar y las coordenadas se envían una a una al NXT por Bluetooth. Cuando las ha recibido todas realiza el recorrido y cuando llega a su posición final, se orienta según la orientación del señuelo final y termina la aplicación.

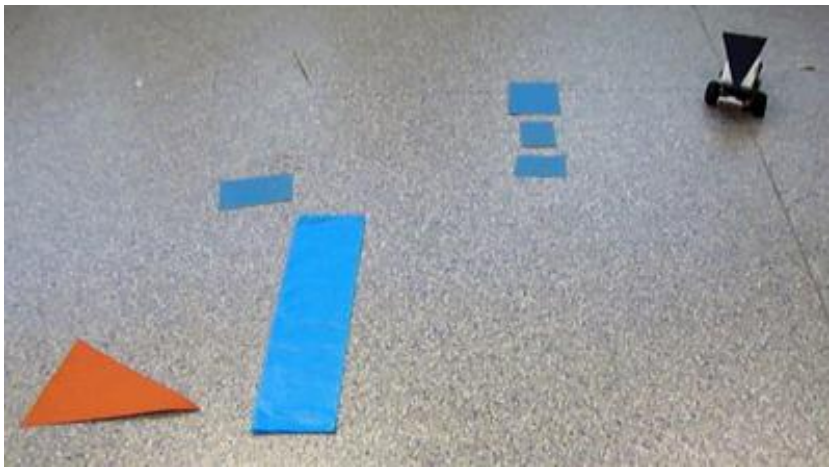


Figura 103. Inicio recorrido

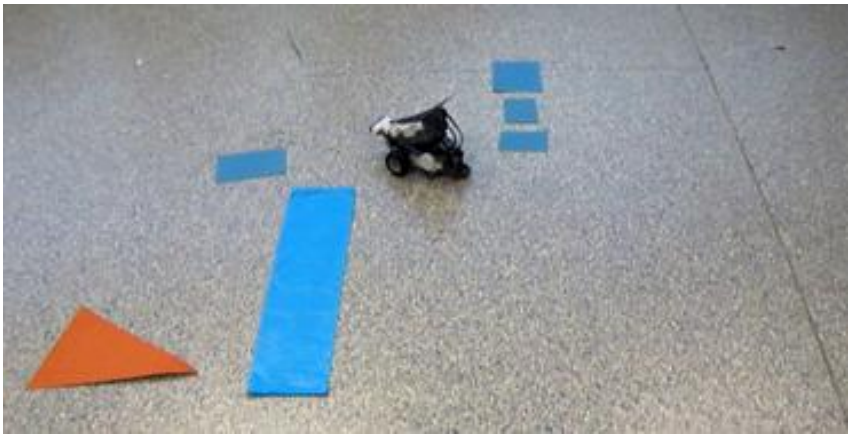


Figura 104. Recorrido

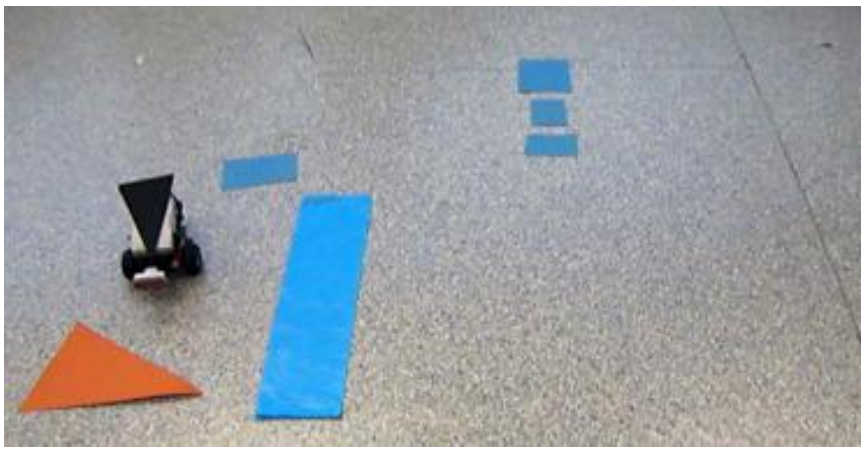


Figura 105. Llegando al objetivo.

Se adjunta un vídeo demostrativo con varios escenarios con las que se realizaron las pruebas.

3.10.4. Problemas encontrados y soluciones

En un primer lugar hay que decir que la cámara cenital con la que se ha trabajado es una Webcam de gama media, por lo que los tiempos de refresco no son excesivamente altos y la calidad de la imagen es medianamente decente. Pero una cámara de alta gama ofrecería unas prestaciones que perfeccionarían el reconocimiento y la discretización que realiza la aplicación.

Prácticamente en todas las sesiones se ha ido calibrando la cámara según la intensidad de luz que había en ese momento en el escenario. Lo ideal sería usar un escenario lejos de ventanas donde la luz diurna no influyese en la percepción de los colores.

Para la generación de trayectorias hay que tener muy en cuenta el tamaño del raster que se debe adecuar a la mínima área que ocuparía el robot en píxeles, ya

que si no, el número de divisiones en las que se descompone la imagen será menor, de este modo el tamaño discretizado de los obstáculos será mayor y por consiguiente será más complicado que el algoritmo encuentre una solución.

De manera parecida ocurre al situar los obstáculos, hay que tener en cuenta que si no se sitúan orientados al eje de coordenadas, ocuparán más coordenadas y será más difícil la generación de trayectorias.

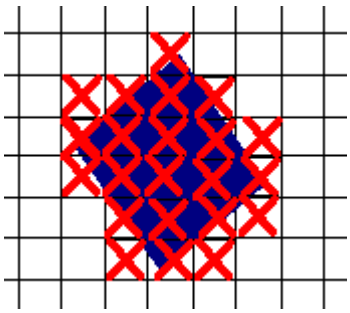


Figura 106. Obstáculo no orientado (22 ocupados)

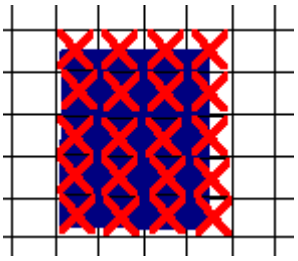


Figura 107. Obstáculo orientado (20 ocupados)

Por otro lado conseguir la orientación final también resultó complicado. Al principio se intentó usando la misma estrategia del punto descentralizado pero añadiendo unos puntos finales a la trayectoria. Estos puntos se calculaban como coordenadas cercanas a la posición final indicada por el señuelo y cuyo ángulo se correspondía con el ángulo final. Es decir, si por ejemplo el robot llegaba al señuelo con una orientación de 45 grados y debía estar en 110, se calculaba un punto cercano al centroide del señuelo que formase un ángulo de 110 respecto a esa posición y se añadía al final de la trayectoria.

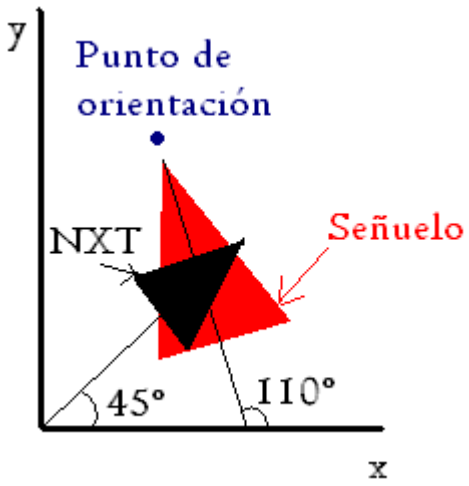


Figura 108. Punto de orientación con NXT bien colocado.

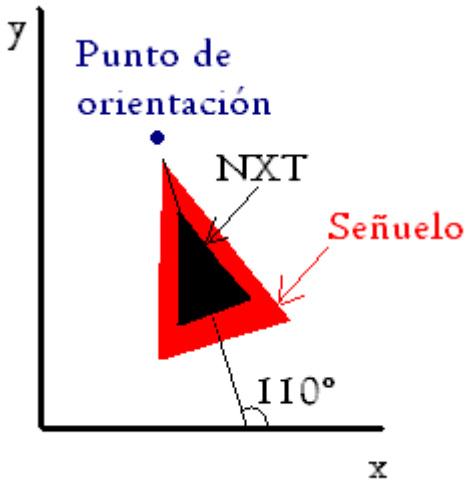


Figura 109. NXT Orientado

Se le otorgaba el suficiente tiempo de muestreo para poder alcanzar la orientación más alejada, y respondía medianamente bien. El problema era que si por cualquier motivo el robot no alcanzaba de manera precisa el centroide del señuelo la orientación en ocasiones no se correspondía con la que debía ser.

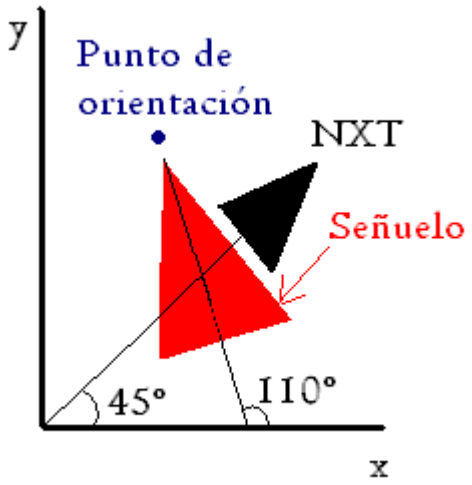


Figura 110. Punto de orientación con NXT de posición final imprecisa.

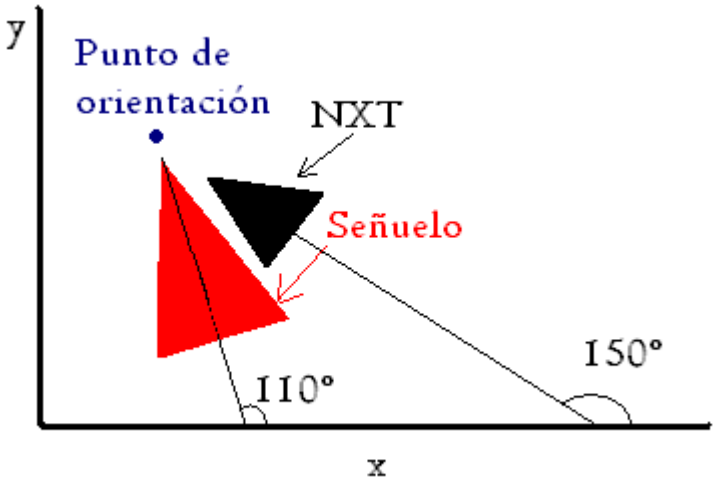


Figura 111. Orientación final errónea.

Se decidió abordar el problema partiendo desde la posición final real que alcanzaba el robot. De esta manera en lugar de operar de manera premeditada desde el PC, se añadió una función de orientación final en el programa del propio robot, la cual bajo la misma estrategia añadía unas coordenadas cercanas a la posición actual del robot, y con una orientación correspondiente a la final. De esta manera, en los casos donde la precisión no era la suficiente en la posición final, el robot se orientaba correctamente.

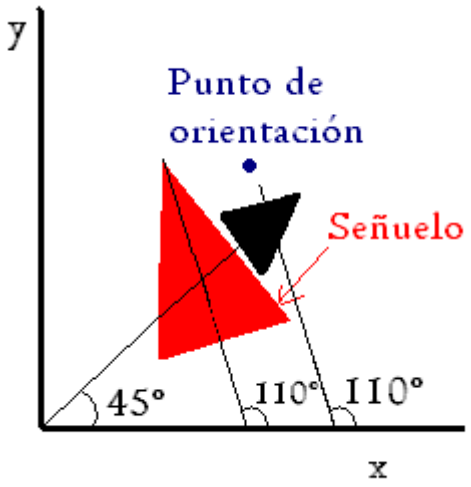


Figura 112. Nuevo punto de orientación.

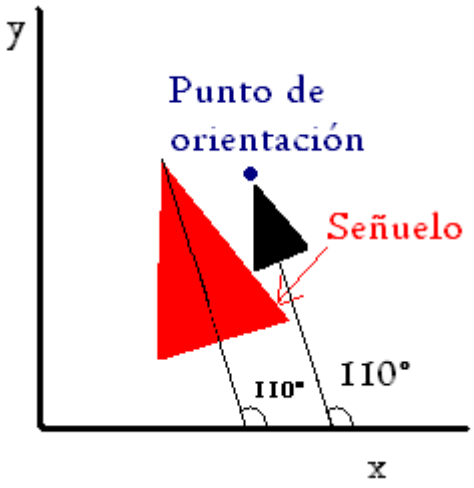


Figura 113. NXT bien orientado.

Aunque el número de resultados satisfactorios aumentó, el problema general residía en el tiempo de muestreo de la estrategia del punto descentralizado. Utilizando esta estrategia para los puntos de orientación no era posible conocer qué periodo de muestreo utilizar ya que para los casos donde el NXT estaba muy mal orientado, se necesitaba más tiempo para alcanzar la orientación final, y para los casos donde prácticamente estaba orientado no se podía dar más tiempo del necesario ya que se salía de su posición.

Por lo que finalmente se optó por el uso de las funciones de alto nivel proporcionadas por LeJOS. Cuando el robot llega a su posición final abandona la estrategia del punto descentralizado, calcula y transforma la orientación que debería alcanzar y hace uso de la función de rotación: rotate() para situarse correctamente.

4.-Conclusiones y futuros proyectos

Se han cumplido con éxito los objetivos que se proponían al comienzo de este proyecto. Se ha alcanzado una gran familiaridad con el firmware LeJOS y con los robots NXT. Se ha conocido a fondo las características de Eclipse como entorno de programación y como herramienta de depuración de código. Se ha estudiado la gestión de la comunicación Bluetooth que se crea del NXT al PC y viceversa, los tiempos de envío de paquetes, las limitaciones, los tiempos de espera… Se han ampliado conocimientos sobre el manejo de varios hilos de ejecución en Java, la herencia entre clases, la gestión de variables compartidas,… Se ha investigado sobre las distintas opciones y librerías que se ofrecen para el tratamiento de imágenes en Java, su discretización, cálculo de centroides, perímetros,.. Se ha trabajado en la gestión y generación de trayectorias de objetos móviles, analizado varias estrategias, calibrado los tiempos de muestreo, buscado los filtros adecuados, …

En definitiva, la tarea de investigar y estudiar los diferentes campos que se proponían se ha visto recompensada y concluida en aplicaciones que cubren con altas expectativas lo estudiado. Tanto la aplicación The Ring, como El laberinto, hacen uso de la comunicación Bluetooth para tratar de manera distinta la generación y gestión de trayectorias. Los problemas y complicaciones surgidos durante la realización del proyecto, han derivado en soluciones que han motivado la ambición de seguir trabajando e investigando, aún después de haber terminado con los objetivos.

Para futuras investigaciones se propone por ejemplo ampliar la aplicación del laberinto para que trabaje en tiempo real generando la trayectoria y así poder introducir en el escenario objetos móviles e incluso más robots NXT. Esto requeriría que el análisis de la imagen capturada por la cámara fuese extremadamente eficiente así como la generación de trayectoria a partir del análisis de los obstáculos.

En el campo de las comunicaciones se propone por ejemplo estudiar la capacidad de utilizar varios computadores con diferentes arquitecturas de conexión para comunicarse entre ellos y entre distintos robots. Por ejemplo, la situación de tener tres ordenadores que se comunican por red y uno esté conectado vía Bluetooth con un Lego Mindstorms NXT o varios, otro a un irobot Create por wifi y otro a un Koala mediante wifi.

También por ejemplo, se propone una aplicación que haga uso de una Scatternet para reconocer los obstáculos de un escenario que debe cruzar un robot móvil de cierto tamaño. El PC se conecta por Bluetooth a un robot situado en una coordenada conocida del escenario y le envía la coordenada destino donde debe dirigirse, pero dicho robot está conectado vía Bluetooth a una red de robots de menores

dimensiones, de la cual es el servidor. Envía la coordenada a sus clientes, y los robots pequeños recorren el escenario detectando las coordenadas que están ocupadas por obstáculos y enviándoselas al robot servidor grande. Cuando encuentran un camino que permite que el robot grande alcance su coordenada destino, los robots pequeños se retiran y el robot servidor recorre el escenario hasta su objetivo teniendo en cuenta las coordenadas ocupadas que se le han enviado.

5.-Bibliografía

[1] Oracle Java. http://www.oracle.com/technetwork/java/index.html

[2] LeJOS: Java for Lego Mindstorms. http://lejos.sourceforge.net/

[3] Wikipedia. http://es.wikipedia.org/wiki/Wikipedia:Portada

[4] Proyectos prácticos de electrónica y robótica.. Jorge Flores Vergaray. http://jorgefloresvergaray.blogspot.com/2009/01/sensores-para-robotica.html

[5] Complubot. De educa Madrid de la universidad de Alcalá de Enares. http://complubot.educa.madrid.org/pruebas/lego_nxt_version_educativa/lego_nxt_version_educativa_index.php

[6] OPENCV: Processing and Java Library. http://www.ubaa.net/shared/processing/opencv/

[7] Christoph Bartneck, PH.D. http://www.bartneck.de/2008/03/04/java-lego-nxt-eclipse-tutorial/#installJava

[8] Industrial Consultant Services. Juan Antonio. http://www.juanantonio.info/jab_cms.php?id=69

[9] Forums nxtasy.org. http://forums.nxtasy.org/lofiversion/index.php/

[10] LEGO Lab, University of Aarhus. http://legolab.daimi.au.dk/

[11] Compile time error messages. http://mindprod.com/jgloss/runerrormessages.html

[12] Bluehack. The Spanish Bluetooth Security Group. http://bluehack.elhacker.net/proyectos/bluesec/bluesec.html

[13] Revista robotiker-tecnalia: Tecnología Bluetooth: características de enlace de radio y establecimiento de la conexión. http://www.robotiker.com/revista/

[14] leJOS Forums, Java for LEGO Mindstorms. http://lejos.sourceforge.net/forum/

[15] Anonimo Colectivo. Taller OpenCV. http://www.anonimocolectivo.org/taller/blog/

[16] Introduction to programming with OpenCV. Gady Adam. Department of Computer Science Illinois Institute of Technology. http://www.cs.iit.edu/~agam/cs512/lect-notes/opencv-intro/index.html

[17] Blog walkintothefuture. OpenCV+Java+linux. http://walkintothefuture.blogspot.com/2009/04/opencv-java-linux.html

[18] Tutorial OpenCV. Raúl Igual. Carlos Medrano. http://www.scribd.com/doc/19006072/Tutorial-Opencv

[19] Manipulación de imágenes TIFF con Java JAI. Miguel Angel Mena Sevilla. http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=manipTIFF

[20] Utilizando JAI api para el Tratamiento de imágenes digitales en JAVA. Duvallier. http://xromsystem.net/?p=1101

[21] Metodo de dilatación de imágenes en Java. Duvallier. http://xromsystem.net/?p=995

[22] Java Advanced Imaging Binary Builds. https://jai.dev.java.net/binary-builds.html

[23] API Java Advanced Imaging. http://download.oracle.com/docs/cd/E17802_01/products/products/java-media/jai/forDevelopers/jai-apidocs/index.html

[24] Java source code. JDK Modules. JAI. http://www.java2s.com/Open-Source/Java-Document/6.0-JDK-Modules/Java-Advanced-Imaging/javax/media/jai/operator/ErodeDescriptor.java.htm

[25] Introducción a OpenCV. Jhon Alvarez Borja. http://my.opera.com/jalvarezborja/blog/introduccion-a-opencv

[26] Sistemas Robotizados. Universidad de Almería. http://aer.ual.es/docencia_es/sr/

[27] Microbótica: Robot Aided Process. http://www.microbotica.es/wiki/doku.php

Anexo 1: Creación de los hilos de ejecución y variables compartidas.

Java ofrece muchos recursos para el uso de varios hilos de ejecución simultáneos. A la hora de crearlos existen básicamente dos opciones: implementar una interfaz de tipo de objeto Runnable o crear una clase que extienda de la clase Thread.

Para todas aplicaciones de este proyecto se ha utilizado la implementación de la interfaz para así aprovechar los métodos definidos en la clase Thread y por simplificación de código.

En la aplicación de The Ring por ejemplo, el hilo que lee las posiciones alcanzadas que envía el NXT y las dibuja en el simulador, se implementa como sigue:

```
public class MiClasehilo extends ConexionaNXT implements Runnable {
	boolean continuar=true;
	FuncionesdeHilos Func;
	public MiClasehilo(FuncionesdeHilos F){
		Func=F;
	}
		public void muere(){
			continuar=false;
		}
		public void run() {
			while(continuar){
				Func.leer();
				Func.dibujar();
				try {
					Thread.sleep(1000);
				} catch (InterruptedException e) {
					e.printStackTrace();
				}

		}
			continuar=true;
		}

}
```

La estructura del manejo de la ejecución de un hilo debe ser siempre parecida, el hilo debe ejecutar un bucle infinito hasta que no se cumpla la condición, salga y se destruya. En este caso, desde la clase Principal, se crea una instancia de la clase MiClasehilo:

```
MiClasehilo HiloLector1 = new MiClasehilo(FdeH);
```

Luego se crea un objeto de tipo hilo al cual se le asigna la interfaz que se ha implementado.

```
Thread hilo1 = new Thread(HiloLector1)
```

Y por último para crear el hilo se ejecuta:

```
hilo1.start();
```

El método start, crea el hilo desde la clase principal e inmediatamente ejecuta el método run() asignado a ese hilo. En este caso, el hilo ejecutará las funciones de leer(), luego dibujar() y se dormirá durante un segundo. Y así continuará hasta que desde la clase Principal donde se ha creado, se llame al método muere(), que hará que la condición del bucle deje de ser cierta y el hilo terminará su ejecución.

En ocasiones como en la aplicación del simulador del apartado 3.4, la vida de un hilo depende directamente de la destrucción de otro. En este caso en concreto sucede al terminar la aplicación mediante una orden de comandos por consola. Del análisis del teclado se encarga un hilo que constantemente escucha lo que se introduce por teclado. En caso de que se introduzca la palabra clave “fin”, el hilo muere y el programa debe terminar. Para ello, el hilo principal se encuentra pausado e intermitentemente va comprobando si el hilo que escucha el teclado sigue vivo:

```
while(HiloEscuchaTeclado.isAlive()){
        try {
                Thread.sleep(1000);
                } catch (InterruptedException e) {
                        e.printStackTrace();
                }
        }
}
findeprograma()
```

Por otro lado, fue necesario también el uso de variables compartidas entre hilos, por ejemplo en la ejecución en los NXT de la aplicación The Ring. Un hilo se encarga de mantener la conexión y mientras tanto se ejecuta un hilo Motor que hace girar al robot y un hilo que escucha intermitentemente al PC por si es necesario cambiar el radio o el centro de la trayectoria que sigue el hilo Motor. En tal caso, el hilo que escucha debe modificar en tiempo real las variables de radio y centro que el hilo Motor está utilizando en ese mismo momento. Para ello se decidió que todos los métodos de los hilos formasen parte de una misma clase llamada FuncionesdeHilos(). De esta clase se crea una instancia en el hilo principal, antes de crear los hilos secundarios a los cuales se les envía la misma instancia de la clase de funciones como parámetro.

```
FuncionesdeHilos FdeH;

FdeH=new FuncionesdeHilos(dis1,dos1,dis2,dos2,dis3,dos3);

MiClasehilo1 HiloLector = new MiClasehilo1(FdeH);
MiClasehilo2 HiloMotor = new MiClasehilo2(FdeH);
```

De este modo, todos los hilos comparten la misma instancia de la clase de métodos que ejecutan y cualquiera puede modificar los valores de variables globales de la clase, que se verán afectadas en la ejecución de todos los métodos que la comparten.

Anexo2: Establecer una comunicación Bluetooth entre el PC y los robots NXT

Para establecer la comunicación Bluetooth entre el PC y los robots NXT se ha creado una clase con el nombre ConexionaNXT() que implementa los métodos conectar() y desconectar().

El método conectar() comienza definiendo el tipo de conexión que se va a iniciar:

```
NXTComm nxtComm = NXTCommFactory.createNXTComm(NXTCommFactory.BLUETOOTH);
```

Y describiendo también cada uno de los robots a los que se pretende conectar:

```
NXTInfo nxtInfo1 = new NXTInfo(NXTCommFactory.BLUETOOTH,"IP4", "00:16:53:08:9d:a5");

NXTInfo nxtInfo2 = new NXTInfo(NXTCommFactory.BLUETOOTH,"NXT", "00:16:53:07:73:fe");

NXTInfo nxtInfo3 = new NXTInfo(NXTCommFactory.BLUETOOTH,"hal9000",
"00:16:53:03:f4:74");
```

También es posible realizar una búsqueda de dispositivos mediante:

```
NXTInfo[] nxtInfo = nxtComm.search(null, NXTCommFactory.BLUETOOTH);
```

Pero los tiempos de espera son superiores, evidentemente, por lo que en todos los casos se ha optado por la opción de describir los dispositivos como en el primer caso.

Ya es posible entonces abrir la comunicación:

```
boolean opened = nxtComm1.open(nxtInfo1);

 if (!opened) {
            System.out.println("Failed to open " + nxtInfo1.name);
            System.exit(1);
}

 boolean opened2 = nxtComm2.open(nxtInfo2);

 if (!opened2) {
            System.out.println("Failed to open " + nxtInfo2.name);
            System.exit(1);
}
```

```
boolean opened3 = nxtComm3.open(nxtInfo3);

if (!opened3) {
        System.out.println("Failed to open " + nxtInfo3.name);
        System.exit(1);
        }

 System.out.println("Connected to all");
```

Y si todo funciona correctamente hay que crear un buffer de escritura y uno de lectura por cada conexión que se realice.

```
InputStream is1 = nxtComm1.getInputStream();
OutputStream os1 = nxtComm1.getOutputStream();

DataOutputStream dos1 = new DataOutputStream(os1);
DataInputStream dis1 = new DataInputStream(is1);

InputStream is2 = nxtComm2.getInputStream();
OutputStream os2 = nxtComm2.getOutputStream();

DataOutputStream dos2 = new DataOutputStream(os2);
DataInputStream dis2 = new DataInputStream(is2);

InputStream is3 = nxtComm3.getInputStream();
OutputStream os3 = nxtComm3.getOutputStream();

DataOutputStream dos3 = new DataOutputStream(os3);
DataInputStream dis3 = new DataInputStream(is3);
```

Y ya estarán los tres NXT conectados al PC. Para enviar un booleano a hall9000 (que es el tercero), por ejemplo, hay tan sólo que acceder al buffer de escritura dos3 con el método correspondiente:

```
dos3.writeBoolean(true);
```

Es importante limpiar también el buffer de escritura para que el receptor sepa cuándo acaba el envío que está recibiendo y para ello se usa el método flush().

En cuando al método implementado desconectar(), hace uso de los métodos close() proporcionados por LeJOS para cerrar la comunicación con los robots conectados. La implementación es como sigue:

```
public void desconecta() throws NXTCommException{
        try {
                dis1.close();
                nxtComm1.close();
                dis2.close();
                nxtComm2.close();
                dis3.close();
                nxtComm3.close();

         } catch (IOException ioe) {
                System.out.println("IOException closing connection");
         }
}
```

Anexo3: Tratamiento de imágenes con OpenCV para Java

Para hacer uso de esta librería tan sólo es necesario descargarla desde http://ubaa.net/shared/processing/opencv/ e incluir la librería OpenCV.jar en el Java Build Path del proyecto.

Se deberá importar hipermedia.video.* para el uso de los métodos de la librería y declarar un objeto de tipo OpenCV() para el acceso a sus métodos.

```
Import hypermedia.video.*;
Constructor(){
        OpenCV cv = new OpenCV();
        cv.capture(  320, 240  ); // el ancho,alto de la ventana
}
```

La función encargada de capturar debe ser un bucle de lectura constante de imágenes capturadas, para lo que se hace uso del método read() y para el tratamiento de la imagen se ha utilizado el método invert(), el cual invierte la imagen, y el método threshold(valor), el cual actúa como un filtro de píxels según el valor del parámetro de entrada. Si el valor es 80 por ejemplo, una llamada a threshold(80), filtrará la salida de los píxeles con valor mayor que 80 hasta 255.

```
while(cv!=null && !parar){
        // grab image from video stream
        cv.read();
        cv.invert();
        cv.threshold( 185 );
```

OpenCV ofrece también un método de reconocimiento de burbujas u objetos que se ha utilizado en el desarrollo de este proyecto. El método recibe un primer parámetro donde se debe indicar el área mínima que debe tener el objeto para ser reconocido como tal, otro parámetro con el área máxima, el máximo de objetos, si se desea el reconocimiento de agujeros y el número máximo de vértices por objeto.

```
Blob[] blobs = cv.blobs( 100, cv.width*cv.height/2, 100, true, OpenCV.MAX_VERTICES*4 );
```

La lista devuelta contiene todos los objetos y de cada objeto, todos sus puntos y píxeles.

```
for( int i=0; i<blobs.length; i++) {

        if(blobs[i].isHole==false){ //si no es agujero
        //Recorrer cada objeto
```

Para pintar los objetos se utiliza la clase Polygon la cual recibe uno a uno los puntos y luego se añaden al gráfico.

```
Polygon shape = new Polygon();
        for( int j=0; j<blobs[i].points.length; j++ ) {
                shape.addPoint( blobs[i].points[j].x, blobs[i].points[j].y );
        }

g.fillPolygon(shape).
```

El análisis de los colores de los píxeles, se lleva a cabo a partir del color del píxel que forma parte del centroide del objeto en cuestión. Según el color se clasifican los objetos distinguiéndolos entre obstáculo, robot y destino.

```
//Análisis del color
        Int tempverde = (ImagenEnbuffer.getRGB(blobs[i].centroid.x,blobs[i].centroid.y)&
0x0000ff00) >>8;
        Int temprojo = (ImagenEnbuffer.getRGB(blobs[i].centroid.x,blobs[i].centroid.y)& 0x00ff0000)
>>16;
        Int tempazul = (ImagenEnbuffer.getRGB(blobs[i].centroid.x,blobs[i].centroid.y)& 0x000000ff);
```

Cuando ya se detectan todos los objetos se calcula la orientación inicial del robot y del señuelo destino. Para ello se analiza cada punto que forma el área y se calcula el punto más alejado al centroide y se dibuja la recta que los une para facilitar el reconocimiento visual del ángulo.

```
double distancia=0;

 int masalejadoR=0; //índice del punto más alejado al centroide del robot

//Cálculo de la orientación del robot

 for(int j=0;j<blobs[indiceRobot].points.length;j++) //para cada punto que forma el área

        if(blobs[indiceRobot].points[j].distance(centroide[indiceRobot].x,
centroide[indiceRobot].y)>distancia){
```

```
                    distancia=blobs[indiceRobot].points[j].distance(centroide[indiceRobot].x,
centroide[indiceRobot].y);

                    masalejadoR=j;

                    }

                //se pinta la recta

                g.drawLine(centroide[indiceRobot].x,          centroide[indiceRobot].y,
blobs[indiceRobot].points[masalejadoR].x, blobs[indiceRobot].points[masalejadoR].y);

                PosInicialX=centroide[indiceRobot].x;

                PosInicialY=centroide[indiceRobot].y;
```

Para calcular el ángulo tan sólo habrá que calcular el arco coseno de la distancia del centroide a la coordenada x del punto más alejado dividido entre la distancia directa del centroide al punto más alejado. Es decir, el arco coseno del cateto adyacente al ángulo entre la hipotenusa.

```
        AnguloInicial=Math.acos((centroide[indiceRobot].distance(blobs[indiceRobot].points[masalej
adoR].x,centroide[indiceRobot].y))/(centroide[indiceRobot].distance(blobs[indiceRobot].points[masal
ejadoR])));
```